Universidad
Carlos III de Madrid

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

BLUEPILLS: ENVÍO DE PÍLDORAS DOCENTES A TRAVÉS DE BLUETOOTH

Autor: Jorge Beca Baulenas
Tutor: Mario Muñoz Organero

Leganés, 6 de julio de 2011

Agradecimientos

A mis amigos, por instarme a recorrer los últimos metros de este largo viaje.

A mis hermanos, mis tías y mi prima, por acompañarme en el trayecto.

A mi tutor Mario, por allanarme el camino, gracias a su enorme amabilidad.

Y a mis padres, por todo.

Resumen

El objeto del presente proyecto es el estudio de un sistema, dentro del ámbito docente, que permita la transmisión de ficheros desde un ordenador a un conjunto de terminales móviles cercanos.

Utilizando la librería *BlueCove* para Bluetooth, se ha implementado una solución software al problema, desarrollada en el lenguaje de programación Java. Aprovechando la versatilidad de Java en el desarrollo de aplicaciones gráficas multiplataforma, se ha buscado una solución que, además de satisfacer los requisitos del problema, facilite su uso.

A pesar de presentar una velocidad de transmisión inferior a otros sistemas de comunicación inalámbrica, Bluetooth posee una serie de características que justifican su uso en el problema propuesto: no requiere configuraciones de red previas y ofrece mecanismos de alto nivel para el intercambio rápido de ficheros.

Palabras clave:

Bluetooth, OBEX, Java, BlueCove, Píldoras docentes.

Abstract

The purpose of this project is the study of a system, within the teaching field, that allows the transmission of files from one computer to a set of mobile terminals nearby.

Using Bluetooth *BlueCove* library, we have implemented a software solution to the problem, developed in the Java programming language. Taking advantage of the versatility of Java in developing cross-platform graphical applications, we have sought a solution that not only satisfies the requirements of the problem, but eases to use.

In spite of having a transmission rate lower than other wireless communication systems, Bluetooth has a number of features that justify its use in the proposed problem: it doesn't require network pre-configurations and provides high-level mechanisms for the rapid exchange of files.

Keywords:

Bluetooth, OBEX, Java, BlueCove, Teaching pills.

Índice general

Capítulo 1

Introducción y objetivos

1.1. Motivación

Hoy en día, más del 90 % de los terminales móviles del mercado incorporan la tecnología Bluetooth. Además, existen numerosas aplicaciones y servicios que permiten comunicar el dispositivo móvil con otros dispositivos Bluetooth para el envío de voz o datos. Estas comunicaciones Bluetooth pueden darse no sólo entre dispositivos móviles, sino también entre muchos otros dispositivos: mandos inalámbricos, ordenadores, impresoras...

Por otro lado, la funcionalidad que ofrecen los terminales móviles es cada día mayor, por lo que empieza a ser habitual que los usuarios utilicen sus teléfonos móviles para nuevos usos, además de para la comunicación de voz. Algunos de estos nuevos usos son: reproductor de música, cámara de fotos, agenda personal, almacenamiento de datos... Lo que aproxima a los móviles cada vez más a los ordenadores personales, con la ventaja de ser portátiles en todo momento. Además, los usuarios pueden comunicar sus móviles y ordenadores personales para sincronizar agendas o transferir archivos. Esta comunicación puede realizarse mediante cables o mediante comunicación inalámbrica, ya sea a través de infrarrojos, Bluetooth o incluso vía Wi-Fi.

En entornos docentes o en aquéllos donde se imparte una charla o curso, el uso de material multimedia constituye hoy una práctica fundamental e indispensable. Este material, dividido en pequeños ficheros o "píldoras" docentes (presentaciones, apuntes, vídeos de corta duración, ejercicios...) no sólo refuerza la práctica docente, sino que permite que los asistentes puedan seguir utilizándolos una vez concluida la sesión, ya sea para repasar, afianzar o ampliar temas y conceptos. Existen muchas maneras de transmitir estas píldoras docentes a los asistentes,

siendo el correo electrónico la más habitual de ellas. Sin embargo, el hecho de que la tecnología Bluetooth esté extendida a todos los móviles, unido a la gran capacidad de almacenamiento que éstos ofrecen, hace que resulte útil poder enviar dichos archivos desde el ordenador que utiliza el orador a los dispositivos móviles de los oyentes, de una manera rápida y sencilla. Gracias a la tecnología Java, es posible desarrollar aplicaciones que hagan esto posible y que, además, se adapten a las necesidades del problema.

1.2. Objetivos

El objetivo principal del presente proyecto es el desarrollo de un sistema que permita enviar un conjunto de ficheros desde un ordenador a una serie de dispositivos móviles, mediante comunicación Bluetooth. A su vez, el sistema debe presentar una serie de características, las cuales podemos enmarcarlas como subobjetivos a perseguir:

1. El sistema debe funcionar en el mayor número posible de dispositivos móviles del mercado, independientemente de la marca y el modelo.

2. El sistema debe ser lo más transparente posible a los usuarios finales de los dispositivos móviles, buscando la mayor sencillez en su uso.

3. La aplicación que realiza el envío de ficheros debe ser sencilla de utilizar y presentar un interfaz amigable al usuario.

4. El sistema debe ser capaz de enviar cualquier tipo de fichero, independientemente del formato.

5. El sistema debe ser capaz de enviar los archivos únicamente a una serie de dispositivos móviles, siguiendo algún determinado criterio.

6. El sistema debe proveer algún mecanismo que permita detectar errores en el envío de los ficheros y ser capaz de realizar un reenvío de los mismos.

7. El sistema debe ser capaz de llevar un seguimiento de los usuarios que han recibido los ficheros, tanto en la sesión actual como en posteriores.

1.3. Fases del desarrollo

El sistema a implementar ha sido desarrollado siguiente una serie de cinco pasos o fases:

1. Desarrollar una aplicación en Java (J2SE) capaz de realizar el envío (vía Bluetooth) de uno o varios ficheros desde un ordenador a los dispositivos móviles circundantes.

2. Desarrollar una aplicación Web en Java (J2EE), que permita el registro al sistema de una serie de usuarios, identificando así su nombre y los datos de su dispositivo móvil. Además, la aplicación Web debe permitir al administrador de la misma consultar, modificar o eliminar los datos de los usuarios registrados.

3. Desarrollar un módulo adicional sobre la aplicación desarrollada en la fase 1, que presente las siguientes funcionalidades adicionales:

   - Muestre un interfaz gráfico al usuario.
   - Permita seleccionar los ficheros a enviar.
   - Permita realizar el envío de ficheros únicamente a los usuarios registrados en la aplicación Web desarrollada en la fase 2.
   - Permita al usuario de la aplicación llevar un seguimiento del estado de los dispositivos móviles circundantes.
   - Implemente un mecanismo que permita llevar un seguimiento de los usuarios que han recibido ficheros en diferentes sesiones, aunque en cada sesión utilicen un terminal móvil distinto.

4. Implementar una mejora adicional en la aplicación Web desarrollada en la fase 2, que permita al usuario inscribirse en uno o más grupos de usuarios.

5. Implementar una mejora adicional en la aplicación desarrollada en la fase 1, que permita a la aplicación enviar los ficheros únicamente a los usuarios de un determinado grupo de usuarios.

1.4. Contenido de la memoria

El contenido de la memoria del proyecto se desglosa en los siguientes capítulos:

- En el Capítulo 2 se describen las tecnologías implicadas en el desarrollo del proyecto: Bluetooth, OBEX, Java y BlueCove.
- El Capítulo 3 muestra la descripción, a alto nivel, de la solución propuesta al problema.
- En el Capítulo 4 se describe el desarrollo de la aplicación Bluepills, analizando el proceso de comunicación entre las distintas entidades del modelo planteado, así como la estructura de módulos y clases Java que lo constituyen.
- En el Capítulo 5 se describe el desarrollo de la aplicación Web Bluepills, complementaria a la aplicación Bluepills.
- En el Capítulo 6 se incluye la batería de pruebas realizadas sobre el sistema completo.
- En el Capítulo 7 se describen las conclusiones extraídas tras la realización del proyecto, así como las posibles líneas de trabajo futuro.
- En el Apéndice A se describe el contenido del presupuesto del proyecto, adjuntado al final del presente documento.
- En el Apéndice B se incluye el manual de instalación y uso de la aplicación Bluepills.
- En el Apéndice C se incluye el manual de instalación y uso de la aplicación Web Bluepills.

Capítulo 2

Estado del arte

2.1. La tecnología Bluetooth

2.1.1. Antecedentes

La tecnología *Bluetooth* es una tecnología de comunicaciones inalámbricas de corto alcance que permite la transmisión de voz y datos entre diferentes tipos de dispositivos, como ordenadores, teléfonos móviles, PDAs, impresoras, auriculares... entre otros muchos.

Bluetooth es el nombre común de la especificación industrial IEEE 802.15.1 para *Redes Inalámbricas de área Personal* (WPANs), la cual define un estándar global de comunicación inalámbrica. Sus orígenes se remontan a 1994, cuando la compañía Ericsson comenzó a investigar distintas alternativas para comunicar sus terminales móviles con ciertos accesorios. Entre muchas de sus conclusiones, determinaron que para que esta nueva tecnología tuviera éxito, resultaba imprescindible que ésta perteneciera a un estándar abierto y no a uno propietario. Para ello, en 1998 formaron el *Bluetooth Special Interest Group* (SIG), junto a otras compañías como Intel, IBM, Nokia o Toshiba. Su objetivo era desarrollar conjuntamente las especificaciones para Bluetooth 1.0, que se terminarían publicando en julio de 1999.

2.1.2. Características radio

Las características que presenta Bluetooth en cuanto a la transmisión radio son las siguientes:

Frecuencia Bluetooth fue diseñado para trabajar en entornos de radio ruidosos, por lo que utiliza un sistema de saltos de frecuencia para garantizar cierta robustez al enlace. Sus módulos radio eliminan la interferencia con otras señales saltando de frecuencia inmediatamente después de transmitir o recibir un paquete.

Bluetooth opera en la banda ISM de frecuencias de 2.4 GHz, por lo que no necesita licencia para su uso. A diferencia de otros sistemas que utilizan la misma banda (como 802.11), Bluetooth realiza saltos de frecuencia más rápidos y emplea paquetes de menor tamaño. La banda de frecuencia de trabajo está dividido en 79 frecuencias diferentes de 1 MHz, entre 2.402 GHz y 2.480 GHz, y pueden realizarse hasta 1600 saltos por segundo.

Transmisión La velocidad de transmisión se encuentra alrededor de 1 Mb/s, un valor inferior a otros estándares radio como 802.11b (11 Mb/s), aunque algo superior a la transmisión infrarroja.

Bluetooth utiliza un esquema de división en el tiempo y una transmisión full-dúplex. Su protocolo de banda base es una combinación de conmutación de paquetes y conmutación de circuitos, y posee cuatro canales: tres canales síncronos de voz y un canal asíncrono de datos. Cada canal de voz permite un enlace síncrono de 64 kb/s. El canal asíncrono permite un enlace asimétrico de 721 kb/s y 57.6 kb/s en la respuesta, o un enlace simétrico de 432.6 kb/s.

Potencia La potencia mínima para funcionar es de 1 mW, un valor considerablemente bajo, lo que permite que dispositivos limitados en potencia puedan utilizarlo sin afectar demasiado al consumo de sus baterías. La potencia máxima permitida según la regulación vigente es de 100 mW (PIRE).

Alcance El estándar Bluetooth define tres clases de transmisores, todos compatibles entre ellos, cuyo alcance varía en función de su potencia radiada (ver cuadro 2.1).

A diferencia de la tecnología infrarroja, que emplea radiación de luz para enviar datos, Bluetooth utiliza ondas de radio a 2.4 GHz. Como consecuencia, los

Clase	Potencia	Alcance
I	100 mW (20 dBm)	100 metros
II	2.5 mW (4 dBm)	10 metros
III	1 mW (0 dBm)	1 metro

Cuadro 2.1: *Clases de dispositivos Bluetooth*

dispositivos no necesitan estar visualmente enfrentados para comunicarse, pudiendo incluso superar obstáculos en el enlace, como pueden ser paredes o ventanas.

Redes Cuando un dispositivo Bluetooth se conecta a otros compartiendo un mismo canal de comunicación, forman una pequeña red denominada *piconet*. Estas redes se componen de un dispositivo maestro que impone la frecuencia de saltos y de uno o más (hasta siete) dispositivos esclavos. No obstante, hasta 255 dispositivos pasivos pueden permanecer conectados a la piconet, de manera inactiva. A su vez, los esclavos pueden estar interconectados a diferentes piconets, formando una *scatternet*. En la figura 2.1 puede verse un ejemplo de la misma.

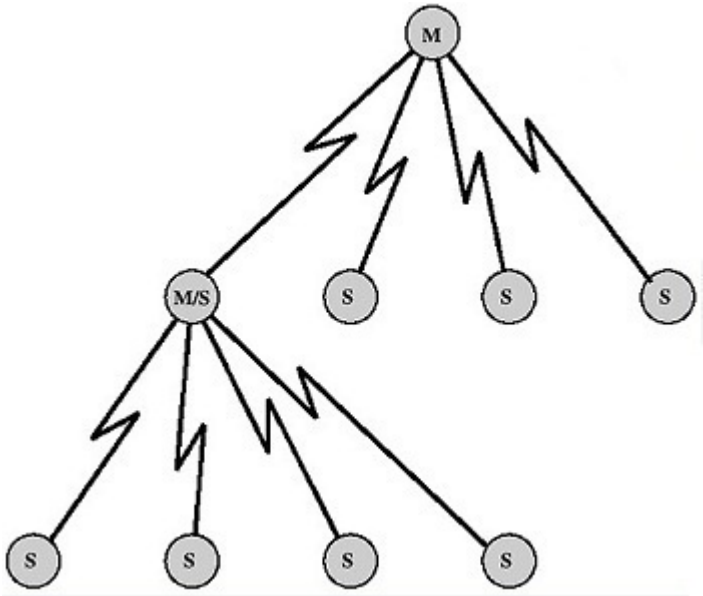


Figura 2.1: *Scatternet Bluetooth*

Una peculiaridad de las redes piconet es que en ellas un dispositivo maestro puede conectarse con cualquier dispositivo esclavo pero, sin embargo, los dispositivos esclavos no pueden comunicarse entre ellos. En lo que a redes scatternet se refiere, un dispositivo puede actuar de esclavo en dos piconets distintas, o actuar

como maestro en una y esclavo en otra.

2.1.3. Versiones

Desde su primera aparición en 1999, han ido surgiendo distintas versiones del estándar Bluetooth, caracterizadas por la introducción de mejoras en cuanto a velocidad de transmisión, consumo de energía y seguridad:

- **Bluetooth v1.1**: El estándar Bluetooth surge a partir de un estudio iniciado por Ericsson para investigar la viabilidad de una nueva interfaz de bajo consumo para la interconexión vía radio entre dispositivos móviles y otros accesorios. Conforme este proyecto fue avanzado, empezó a vislumbrarse que este tipo de enlace podía ser utilizado en un gran número de aplicaciones, al estar basado en único chip de radio.
- **Bluetooth v1.2**: Provee una solución inalámbrica complementaria que permite la coexistencia de Bluetooth y Wi-Fi en el espectro de frecuencias de 2.4 GHz. Utiliza la técnica *Adaptative Frequency Hopping* (AFH), que permite una transmisión más eficiente y un cifrado más seguro. Ofrece además una calidad de voz con menor ruido ambiental y una configuración más rápida con el resto de dispositivos Bluetooth.
- **Bluetooth v2.0**: Incorpora la técnica *Enhanced Data Rate* (EDR), que permite mejorar la velocidad de transmisión hasta 3 Mbps, y corrige algunos errores detectados en la especificación 1.2.
- **Bluetooth v2.1**: Disminuye hasta cinco veces el consumo de potencia y simplifica el proceso de creación de conexiones entre dispositivos.
- **Bluetooth v3.0**: Aumenta considerablemente la velocidad de transferencia.
- **Bluetooth v4.0**: Disminuye drásticamente el consumo de potencia, lo que permite incorporar la tecnología Bluetooth en pequeños dispositivos, como relojes o reproductores portátiles. Se mejora la seguridad, introduciendo cifrado AES-128.

2.1.4. Arquitectura de la pila Bluetooth

La pila de protocolos Bluetooth puede dividirse en dos grandes bloques: el *Bluetooth Host* y el *Bluetooth Controller*. Cada uno de ellos, como puede verse en la figura 2.2, consta de una serie de protocolos, los cuales describiremos a continuación. El *Host Controller Interface* (HCI), situado entre medias, proporciona un interfaz estandarizado entre ambos. El Bluetooth Host generalmente está implementado en software y se encuentra integrado con el sistema operativo del dispositivo. El Bluetooth Controller (también llamado *Bluetooth Radio Module*) es el módulo hardware del dispositivo Bluetooth que interactúa con el Bluetooth Host a través de mecanismos de entrada/salida, como puede ser PCMCIA (*Peripherical Componente Microchannel Interconnect Architecture*), UART (*Universal Asynchronous Receiver-Transmiter*) o USB (*Universal Serial Bus*). No obstante, aunque esta subdivisión de la pila Bluetooth se aplica en la mayoría de dispositivos, en muchos de ellos ambas partes se encuentran integradas, prescindiendo del uso del HCI.

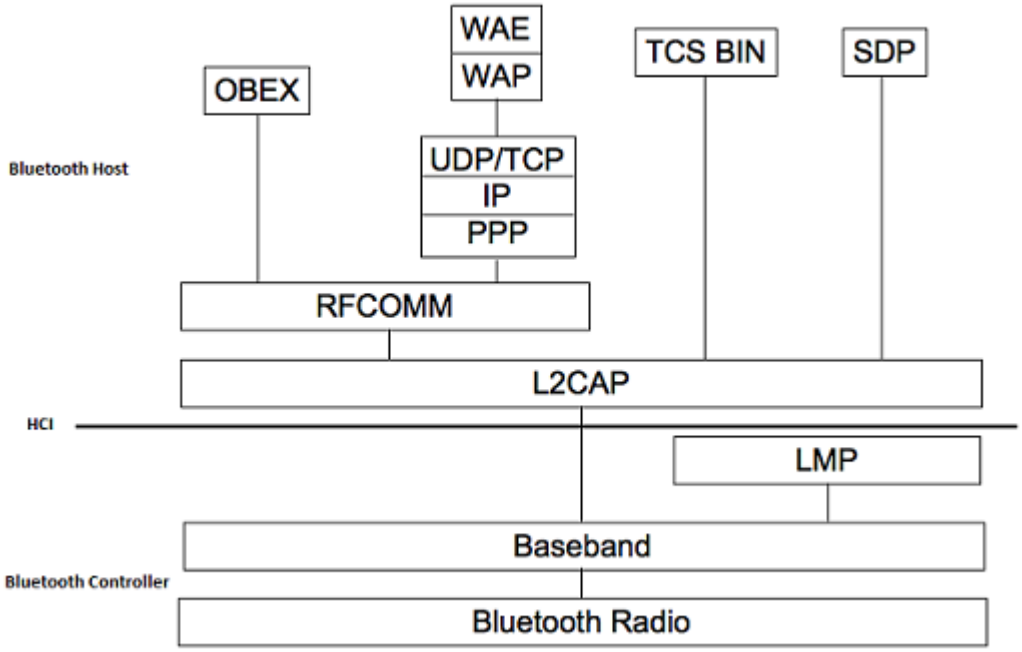


Figura 2.2: *Pila de protocolos Bluetooth*

La pila de protocolos está compuesta por protocolos específicos de la tecnología Bluetooth, como SDP, mientras que otros son protocolos adoptados de otras tecnologías, como es el caso de OBEX. A continuación describimos brevemente la funcionalidad de cada uno de ellos:

- **Bluetooth radio**: La capa de radio Bluetooth es la capa más baja de las definidas en la especificación. Define los requisitos del dispositivo transceptor radio, como son la frecuencia de los canales, los saltos en frecuencia o los niveles de potencia de transmisión.

- **Baseband**: La capa de banda base permite realizar una conexión entre dos dispositivos Bluetooth. Gestiona el procesado de canales y la temporización, así como el acceso a los canales. Permite dos tipos de canales: *Orientado a Conexión Síncrono* (SCO) y *No Orientado a Conexión* (SCL). Un enlace ACL transporta paquetes de datos, mientras que un enlace SCO transporta tráfico de audio en tiempo real. El audio no es en realidad una capa de la pila de protocolos, y se encamina directamente a la capa de banda base sobre un enlace SCO o ACL.

- **Link Manager Protocol (LMP)**: El *Protocolo de Gestión de Enlace* (LMP) es el responsable de gestionar las conexiones entre los dispositivos Bluetooth, con todos los pasos que ello implica. Se encarga además de controlar y negocia el tamaño de los paquetes que se envían al nivel de banda base, así como de los aspectos de seguridad, como son la autenticación y el cifrado de datos.

- **Host Controller Interface (HCI)**: El HCI, comentado anteriormente, es un interfaz estándar para acceder a las capacidades banda base de Bluetooth y a los registros de estado y control del hardware.

- **Logical Link Control and Adaptation Protocol (L2CAP)**: El *Protocolo de Control Lógico de Enlace y Adaptación* (L2CAP) oculta los detalles de los protocolos inferiores a los protocolos de las capas superiores, multiplexando sus conexiones lógicas.

- **Service Discovery Protocol (SDP)**: El *Protocolo de Descubrimiento de Servicios* (SDP) permite a las aplicaciones solicitar qué servicios hay disponibles, así como las características de los mismos. A diferencia de las *Redes de Área Local* (LAN), en donde los dispositivos se conectan primero a la red y luego encuentran otros dispositivos, en el estándar Bluetooth primero encuentran a los dispositivos existentes en la red y, tras descubrir sus servicios, se conectan a ellos. Además, los servicios disponibles pueden cambiar mientras los dispositivos siguen conectados.

- **RFCOMM**: Los puertos serie son uno de los interfaces más comunes de comunicación entre dispositivos. La función del protocolo RFCOMM es emular los puertos serie sobre L2CAP para proporcionar un mecanismo de transporte de datos para servicios de alto nivel. RFCOMM permite realizar

tanto múltiples conexiones concurrentes a un dispositivo como conexiones de uno a múltiples dispositivos.

- **Telephone control Specification (TCS)**: El *Protocolo de Especificación de Control de Telefonía Binario* (TCS Binary) define la señalización de control para el establecimiento de llamadas de voz y datos entre dispositivos Bluetooth.

Otros protocolos como OBEX e IP funcionan sobre alguno de los protocolos descritos anteriormente. La descripción del protocolo OBEX (*OBject EXchange*), en particular, se detallará en la sección 2.2.

2.1.5. Perfiles Bluetooth

Además de los protocolos, el estándar Bluetooth define una serie de perfiles. Un perfil Bluetooth puede describirse como una selección vertical en la pila de protocolos, y define tanto las diferentes capas de protocolos empleados como las distintas características de cada uno de ellos. Un dispositivo Bluetooth puede soportar uno o más perfiles. Los cuatro perfiles básicos son:

- **Generic Access Profile (GAP)** Es el perfil base de los demás perfiles. Define los procedimientos genéricos relativos al establecimiento de conexiones entre dos dispositivos, incluyendo el descubrimiento de dispositivos Bluetooth, la gestión de enlace y la configuración, y los procedimientos relacionados con la seguridad.

- **Service Discovery Application Profile (SDAP)** Describe los protocolos y procedimientos necesarios para descubrir los servicios disponibles en otros dispositivos Bluetooth, utilizando el *Protocolo de Descubrimiento de Servicios* (SDP).

- **Serial Port Profile (SPP)** Define los requisitos necesarios en los dispositivos Bluetooth para emular conexiones serie entre dos terminales usando el protocolo RFCOMM.

- **Generic Object Exchange Protocol (GOEP)** Proporciona un modelo genérico para otros perfiles que implementen el protocolo OBEX y define los papeles para los dispositivos clientes y servidores. Estipula que debe ser el cliente el que inicie la transacción pero, sin embargo, no describe cómo las aplicaciones deberían definir los objetos a intercambiar ni cómo se debería implementar exactamente el intercambio. Esos detalles se dejan

en manos de perfiles OBEX más específicos, como *OBEX File Transfer* u *OBEX Object Push*.

2.1.6. Seguridad en Bluetooth

Bluetooth incorpora varios mecanismos de seguridad, definidos a dos niveles: nivel banda base y nivel de enlace.

Nivel banda base

Está basada en la secuencia de saltos en frecuencia, sólo conocida por los dispositivos pertenecientes a la misma piconet, y establecida a priori por el dispositivo maestro. Proporciona cierta confidencialidad a los datos, siempre y cuando el posible atacante desconozca la secuencia de saltos.

Nivel de enlace

En él se definen tres mecanismos de seguridad:

1. **Autenticación** Cuando dos dispositivos Bluetooth intentan comunicarse, se inicia un procedimiento de emparejamiento o *pairing*, creando una clave de enlace común conocida por ambos dispositivos.

2. **Autorización** El mecanismo de autorización se lleva a cabo mediante niveles de confianza, los cuales determinan la capacidad de acceso de un determinado dispositivo a los servicios ofrecidos por el servidor:

Nivel de confianza	**Pairing**	**Acceso a servicios**
TOTAL	Sí	Sin restricciones
PARCIAL	Sí	Restringido a uno o varios servicios
NULA	No	Restringido a todos los servicios

Cuadro 2.2: *Niveles de confianza*

Todos los dispositivos Bluetooth disponen de una base de datos interna con una lista de dispositivos de confianza y el nivel de confianza asociado a cada uno de ellos. En caso de que un determinado dispositivo no confiable intente acceder a un servicio restringido, se requiere un procedimiento explícito de confirmación por parte del usuario.

3. **Cifrado** El cifrado de datos protege la información transmitida en una conexión Bluetooth, garantizando su confidencialidad. No obstante, su implementación es opcional, por lo que maestro y esclavo deben acordar su uso.

Relacionados con los tres mecanismos de seguridad Bluetooth a nivel de enlace, se definen para cada dispositivo y servicio tres modos de seguridad en función de su grado de implementación:

- **Modo 1:** Todos los mecanismos de seguridad están deshabilitados. Además, el dispositivo funciona en modo *promiscuo*, permitiendo que todos los dispositivos Bluetooth puedan conectarse a él. No se cifran los datos.
- **Modo 2:** Proporciona seguridad a nivel L2CAP. Utiliza únicamente autorización, por lo que la interacción con el usuario se limita a solicitar su confirmación de acceso. Sólo se cifran las conexiones punto a punto.
- **Modo 3:** Proporciona seguridad a nivel LMP. Utiliza autenticación, por lo que requiere el emparejamiento de dispositivos mediante una clave compartida por ambos e introducida por el usuario (código PIN). Se cifran todas las conexiones.

Los primeros terminales que implementaron Bluetooth incorporaban por defecto el Modo 1. Poco a poco y debido al enorme riesgo que esto suponía para la seguridad, los distintos servicios han ido incorporando cada vez modos de seguridad más altos. Hoy en día, prácticamente la totalidad de los teléfonos móviles del mercado incorporan el Modo 3 en todos sus servicios, a excepción del *Perfil de Carga de Objectos* (OBEX Object Push Profile), que utiliza el Modo 2.

En lo que a descubrimiento de dispositivos se refiere, existen dos modos: *Modo Visible* (Discoverable) y *Modo No Visible* (Non Discoverable). No obstante, aunque un dispositivo se encuentre en modo No Visible, es posible realizar conexiones con él si se conoce previamente su *Dirección Física Bluetooth* (Bluetooth Address).

2.2. OBEX

2.2.1. Descripción

OBEX (*OBject EXchange*) es un protocolo de nivel de sesión desarrollado originariamente por la asociación IrDA (*Infrared Data Association*) para el intercambio de objetos de forma sencilla. Su funcionalidad es similar al protocolo HTTP: utiliza mensajes compuestos por una o más cabeceras de mensaje y (opcionalmente) un cuerpo de mensaje. A su vez, los mensajes de respuesta poseen un código de respuesta indicando el éxito o notificando un posible error.

El protocolo OBEX define las distintas operaciones que puede realizar un cliente con el servidor:

- **CONNECT:** Inicia una sesión.
- **DISCONNECT:** Finaliza una sesión.
- **PUT:** Envía un objeto al servidor.
- **GET:** Solicita un objeto al servidor.
- **DELETE:** Solicita la eliminación de un objeto del servidor.
- **SETPATH:** Solicita un cambio del directorio actual dentro del árbol de directorios del servidor.

Como se comentó en la sección 2.1.5, el perfil GOEP incluye dos perfiles más específicos:

- **OBEX Object Push:** Proporciona un escenario de conexión rápida (transferencia de objetos y desconexión), por lo que sólo permite las operaciones CONNECT, PUT GET y DISCONNECT. Utiliza un nivel de seguridad Modo 2, por lo que sólo es necesaria la autorización por parte del usuario del dispositivo servidor para poder establecer la conexión. Esto es debido a que en un principio su uso estaba destinado al intercambio de tarjetas de visita entre dispositivos móviles. Actualmente el perfil conserva esta funcionalidad, pero puede utilizarse también para la transferencia rápida de ficheros. No obstante, debido a su bajo nivel de seguridad, como servicio sólo está implementado en dispositivos móviles.

- **OBEX File Transfer:** Establece sesiones en las que las transferencias tienen lugar durante un largo período de tiempo, manteniendo la conexión incluso cuando ésta esté inactiva. Permite todas las operaciones OBEX y utiliza un nivel de seguridad Modo 3, por lo que es necesario la autenticación de ambas partes, cliente y servidor, mediante el uso de un código PIN conocido y compartido por ambos. Lo implementan todos los dispositivos Bluetooth que gestionan sistemas de ficheros.

Debido a su implementación en el presente proyecto, en la siguiente subsección se detalla el funcionamiento del perfil OBEX Object Push en particular. El esquema del perfil OBEX File Transfer es similar, salvo que permite más operaciones, como se ha descrito en la comparativa anterior.

2.2.2. OBEX Object Push

El *Perfil de Carga de Objetos* (OBEX Object Push) permite la "carga" (push en inglés) de objetos desde un cliente a un servidor OBEX. Estos objetos pueden ser una tarjeta de visita, un fichero o simplemente un texto plano. La especificación del perfil OBEX Object Push [2] define cómo deben comunicarse las aplicaciones con el perfil, pero no define los requisitos para los niveles Baseband, LMP, L2CAP o RFCOMM.

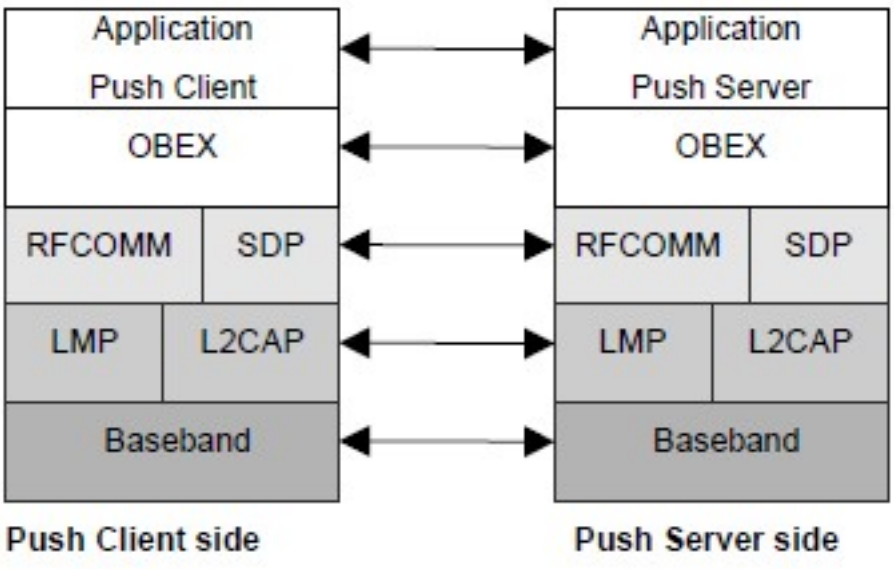


Figura 2.3: *Pila de protocolos para OBEX Object Push*

CLIENTE	**SERVIDOR**
	Activa el modo *Object Exchange*
Realiza una búsqueda del servicio OBEX Object Push en el dispositivo servidor	
Inicia una conexión OBEX y transmite el objeto al servidor	
	El objeto es recibido por el servidor, tras solicitar previamente al usuario si acepta o rechaza el envío
Opcionalmente la aplicación cliente puede informar al usuario del resultado de la operación	

Cuadro 2.3: *Diagrama de comunicación OBEX Object Push*

En la figura 2.3 se muestra la relación entre un cliente y un servidor OBEX Object Push. Como podemos observar, por debajo del nivel de aplicación la pila de protocolos Bluetooth mantiene el esquema descrito en la sección 2.1.4. El protocolo SDP permitiría el descubrimiento, por parte del cliente, del servicio OBEX Object Push ofrecido por el servidor. Por otro lado, hay que tener en cuenta que la conexión siempre debe iniciarla el cliente, ya sea para enviar un objeto (mediante la operación PUT) o para solicitar el envío de un objeto (mediante la operación GET) por parte del servidor.

Para el caso particular del envío (o "push") de objetos (la implementada en el presente proyecto), la secuencia de comunicación que ambas aplicaciones, cliente y servidora, deben seguir es la mostrada en el cuadro 2.3.

En cuanto a cuestiones de seguridad, los requisitos son los mismos que los definidos en el perfil GOEP: la autenticación y el cifrado de datos a nivel de enlace son obligatorias de implementar aunque su uso es opcional.

2.3. Java y Bluetooth

2.3.1. El estándar JSR-82

Los APIs (*Application Programming Interface*) de Java permiten, entre otras cosas, diseñar aplicaciones independientes del hardware, el sistema operativo o el tipo de dispositivo empleado. Además, al ser un lenguaje de alto nivel orientado a objetos, Java permite una gran capacidad de abstracción a la hora de modelar aplicaciones. Por éstas y otras razones, se desarrollaron unos estándares API para Bluetooth utilizando el lenguaje de programación Java, conocidos como JABWT (*Java APIs for Bluetooth Wireless Technology*). En Java, todas las configuraciones, perfiles y paquetes opcionales se definen como un JSR (*Java Specification Request*). En el caso de JABWT, éste se definió como el estándar JSR-82.

El estándar JSR-82 permite esconder la complejidad y los detalles de bajo nivel del estándar Bluetooth y centrarse en el desarrollo rápido y sencillo de aplicaciones. Las capacidades que JSR-82 ofrece son las siguientes:

- Registro de servicios.
- Descubrimiento de dispositivos y servicios.
- Establecer conexiones RFCOMM, L2CAP y OBEX entre dispositivos para el envío de datos. La comunicación de voz, no obstante, no está soportada.
- Establecer seguridad en las comunicaciones, mediante autenticación y cifrado de datos.

Los APIs Java para Bluetooth definen dos paquetes dentro del paquete `javax.microedition.io`: `javax.bluetooth` y `javax.obex`.

2.3.2. BlueCove

En un principio, el estándar JSR-82 fue diseñado exclusivamente para la plataforma J2ME (*Java 2 Micro Edition*), la cual permite desarrollar aplicaciones para dispositivos embebidos, fundamentalmente terminales móviles. No obstante, también es posible diseñar aplicaciones Bluetooth para dispositivos que utilizan J2SE (*Java 2 Standard Edition*), como ordenadores y otros dispositivos de mayor capacidad. Esto es posible mediante la instalación de la librería *BlueCove*.

BlueCove es una librería de J2SE para Bluetooth que posee los interfaces JSR-82 pero, sin embargo, no puede definirse formalmente como una implementación

de JSR-82, ya que actualmente no pasa todos los tests JSR-82 TCK (para más información, véase la referencia [3]). BlueCove posee además licencia LGPL, por lo que es posible distribuir libremente software implementado con BlueCove.

La librería BlueCove permite interactuar con MAC OS X, WIDCOMM, BlueSoleil, Linux (con BlueZ) y la pila Bluetooth de Microsoft. En este último caso, BlueCove funciona en los sistemas operativos *Windows XP Service Pack 2*, *Windows Vista*, *Windows 7* (32bits) y *Windows Mobile 2003* o superior. En lo que a Java se refiere, BlueCove funciona con la versión 1.1 de J2SE y superiores.

En la figura 2.4 se muestra la arquitectura de la librería BlueCove en relación con la pila de protocolos Bluetooth. Como podemos observar, los protocolos de niveles inferiores (Bluetooth Controller) son implementados por el hardware Bluetooth, mientras que la pila Bluetooth Host y el HCI son implementados por el propio sistema operativo. La aplicación en ejecución residiría en la *Máquina Virtual de Java* (JVM), y las librerías BlueCove ofrecerían un interfaz entre ésta y los protocolos de la pila Bluetooth.

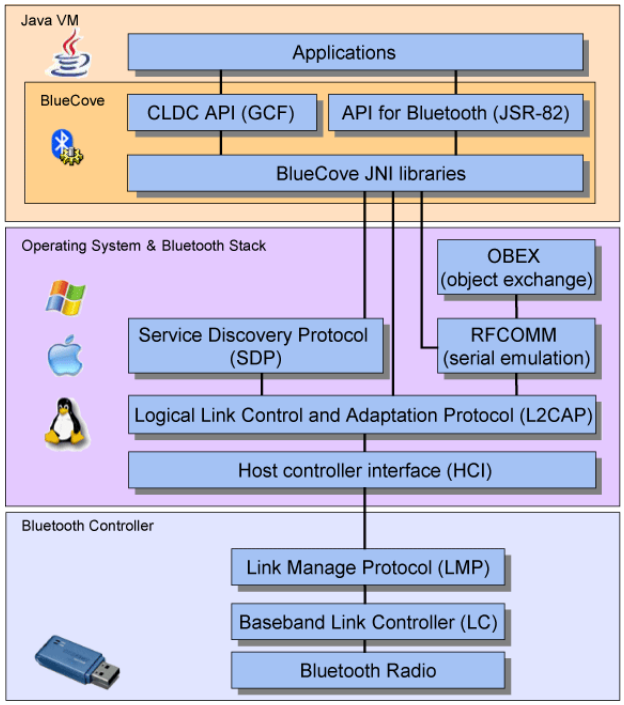


Figura 2.4: *BlueCove y la pila Bluetooth*

2.3.3. El paquete javax.bluetooth de Java

Como se comentó anteriormente, en toda comunicación Bluetooth existe un dispositivo que ofrece un servicio (servidor) y uno o más dispositivos que acceden a él (clientes). A continuación se describen, desde el punto de vista de las aplicaciones Java, las funciones que desempeñan cada uno de ellos, y que pueden implementarse mediante el API `javax.bluetooth` del JSR-82.

CLIENTE BLUETOOTH

Para comunicarse con un servidor Bluetooth, los pasos que la aplicación cliente Bluetooth debe realizar son los siguientes:

1. Búsqueda de dispositivos
2. Búsqueda de servicios en el dispositivo
3. Establecimiento de la conexión
4. Comunicación: transmisión de datos
5. Cierre de la conexión

SERVIDOR BLUETOOTH

A diferencia de las aplicaciones cliente Bluetooth, la aplicación servidora no realiza ninguna búsqueda de dispositivos. Ésta debe tan sólo registrar el servicio que desea ofrecer y esperar a que los clientes se conecten a ella. En este caso, los pasos que la aplicación servidora debe seguir son los siguientes:

1. Crear una conexión servidora
2. Especificar los atributos de servicio
3. Escuchar posibles conexiones cliente
4. Abrir las conexiones cliente

2.3.4. El paquete javax.obex de Java

Como se comentó en la sección 2.2.1, OBEX es un protocolo de alto nivel muy similar a HTTP, ya que se basa en mensajes compuestos por una o más cabeceras de mensaje y, opcionalmente, un cuerpo de mensaje. El cuerpo del mensaje es en realidad un objeto, y éste puede ser un archivo, un texto, una tarjeta de visita, un array de bytes...

Al igual que en HTTP, los mensajes de petición del cliente al servidor se clasifican por métodos (CONNECT, PUT, GET, DELETE, SETPATH y DISCONNECT), y se implementan mediante la clase `ClientSession`. La funcionalidad de estos métodos u operaciones fue descrita anteriormente en la sección 2.2.1.

Las cabeceras o *headers* de un mensaje OBEX se encapsulan en Java mediante un objeto de la clase `HeaderSet`. Existen cabeceras comunes que indican distintas características del mensaje, como son su nombre, su tamaño... aunque también pueden crearse cabeceras personalizadas. Las cabeceras de mensaje más habituales son:

Identificador	**Función**	**Tipo**
COUNT	Número de objetos utilizados en la sesión	`Long`
NAME	Nombre del objeto	`String`
TYPE	Tipo MIME del objeto	`String`
LENGTH	Tamaño del objeto	`Long`
TIME_4_BYTE	Fecha y hora del objeto	`Calendar`
DESCRIPTION	Breve descripción del objeto	`String`

Cuadro 2.4: *Tipos de cabeceras de mensaje OBEX*

Para enviar y recibir mensajes que no sólo poseen cabeceras sino también un cuerpo de mensaje (operaciones GET y PUT), se utiliza la clase `Operation`. A través de un objeto de esta clase y, tras establecerse la conexión entre cliente y servidor, pueden enviarse u obtenerse ficheros mediante flujos de bytes.

2.4. Java GUI: Swing

Swing es una biblioteca gráfica de Java, perteneciente a la plataforma J2SE, que proporciona un rico conjunto de componentes GUI (*Graphical User Interface*), como cajas de texto, botones, menús, desplegables y tablas.

Desde sus inicios, el entorno Java ya contaba con una biblioteca de componentes gráficos, conocida como AWT (*Abstract Window Toolkit*). Esta biblioteca estaba concebida como una API estandarizada que permitía utilizar los componentes nativos de cada sistema operativo. En la práctica, esta tecnología no funcionó como se esperaba, ya que tanto el funcionamiento como el comportamiento de los controles variaba mucho de un plataforma a otra.

Por otro lado, existía ya desde 1996 una biblioteca gráfica para el lenguaje de programación Java, conocida como las *Internet Foundation Classes* (IFC), y desarrollada por Netscape. A diferencia de AWT, los componentes de IFC eran mostrados y controlados directamente por código Java, independientemente de la plataforma. Dichos componentes, denominados componentes ligeros, no requerían reservar recursos nativos del sistema de ventanas del sistema operativo.

En 1997, Sun Microsystems y Netscape Communications Corporation anunciaron su intención de combinar IFC con otras tecnologías de las *Java Foundation Classes*, de cuyo resultado surgió Swing. Además de los componentes ligeros suministrados originalmente por la IFC, Swing introdujo un mecanismo que permitía que el aspecto de cada componente de una aplicación pudiese cambiar sin introducir cambios sustanciales en el código de la aplicación. La introducción de un soporte ensamblable para el aspecto, permitió a Swing emular la apariencia de los componentes nativos manteniendo las ventajas de la independencia de la plataforma.

2.4.1. Características

Incluso el más sencillo de los componentes Swing tiene capacidades que van más allá de las ofrecidas por los componentes AWT. Las principales características que presenta Swing son:

- Está desarrollado totalmente en java.
- No reemplaza a AWT, sino que se apoya sobre él y le añade *JComponents*.
- Añade nuevos componentes, como árboles, tablas y frames internos... así como iconos y bordes.

- Utiliza un modelo basado en eventos.
- Swing no se apoya en los componentes proporcionados por el sistema, por lo que éstos se comportan igual en cualquier plataforma. Por otro lado, la apariencia que presentan (*look and feel*) es configurable.
- Se puede modificar fácilmente el comportamiento o la apariencia de un componente Swing llamando a métodos o creando una subclase.
- Las tecnologías asistidas como los lectores de pantallas pueden fácilmente obtener información desde los componentes Swing. Por ejemplo, una herramienta puede fácilmente obtener el texto mostrado en un botón o en una etiqueta.

2.4.2. Componentes

Toda aplicación gráfica realizada con Swing consta de al menos un contenedor principal donde se sitúan todos los demás componentes. En el caso de aplicaciones con ventanas estándar, el contenedor principal pertenece a la clase `javax.swing.JFrame`. Esta clase representa a la ventana principal de la aplicación y suele contener, al menos: un marco con título y botones para minimizar o cerrar la ventana, una barra de menús, una barra de herramientas con botones...

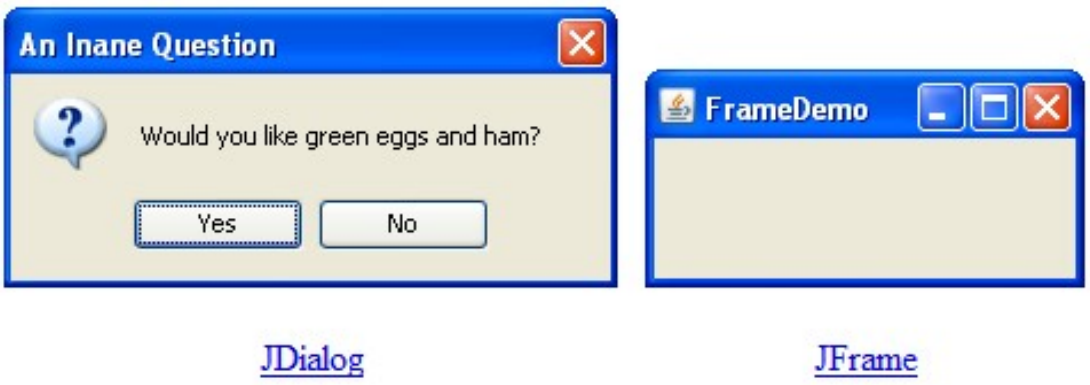


Figura 2.5: *Ejemplo de JFrame y JDialog*

Además de la ventana principal, otro tipo de ventanas secundarias que se suelen utilizar son:

- `JDialog`: Ventana que suele mostrar un mensaje o una pregunta al usuario, y que no se puede minimizar ni maximizar.

- `JFileChooser`: Ventana que permite seleccionar un fichero o un directorio del sistema.

Dentro de cada ventana, pueden añadirse uno o más paneles de distintos tipos:

- `JPanel`: Panel de tamaño fijo.
- `JScrollPane`: Panel con barras de desplazamiento.
- `JTabbedPane`: Grupo de paneles en pestañas.

A su vez, dentro de cada panel pueden añadirse, entre otros, los siguientes componentes gráficos:

- `JButton`: Botón con texto e imágenes.
- `JRadioButton`: Botones de selección única.
- `JCheckBox`: Casillas de selección múltiple.
- `JComboBox`: Botón de selección desplegable.
- `JSpinner`: Casilla de texto numérico, con botones para incrementar o decrementar su valor secuencialmente.
- `JTextArea`: Área de texto editable o no editable.
- `JTable`: Tabla con celdas seleccionables y/o editables.
- `JTree`: Árbol de elementos, por ejemplo, directorios y ficheros.

Con respecto a los elementos de la barra de menús, Swing permite crear:

- `JMenuBar`: Barra de menú.
- `JMenu`: Menú, del que se despliegan uno o más elementos.
- `JMenuItem`: Elemento del menú.
- `JRadioButtonMenuItem`: Elemento de menú de selección única.
- `JCheckboxMenuItem`: Elemento de menú con selección multiple.

Los menús se construyen de forma jerárquica: la barra de menú consta de varios menús, y cada menú consta de varios elementos de menú o submenús (elemento perteneciente también a `JMenuItem`).

2.4.3. Hilos en Swing

Una aplicación Swing bien diseñada usa concurrencia para evitar bloqueos y lograr que ésta responda siempre a la interacción del usuario, independientemente de la circunstancia. Para ello, es importante conocer primero qué tipos de hilos utilizan el *framework* Swing:

- *Initial Threads*: Hilo o hilos que ejecutan el código inicial de la aplicación, y que sitúan y configuran todos los elementos gráficos.
- *Event Dispatch Thread*: Hilo que ejecuta todo el código manejador de eventos; fundamentalmente, los eventos ocurridos cuando el usuario interactúa con alguno de los elementos de la interfaz gráfica.
- *Worker Threads*: También conocidos como *Background Threads* (hilos de tarea de fondo), son hilos que se ejecutan en un segundo plano y que realizan tareas que consumen mucho tiempo.

Como cualquier otro programa de la plataforma Java, un programa Swing puede crear hilos adicionales pero, sin embargo, sólo necesita los hilos descritos anteriormente para funcionar.

En los programas Swing, los hilos iniciales no tienen demasiadas tareas que realizar. Su trabajo esencial es crear un objeto `Runnable` que inicializa la interfaz gráfica y lo programa para su ejecución en el hilo despachador de eventos. Una vez creados los elementos gráficos, la ejecución del programa es conducida principalmente por los acontecimientos ocurridos en la interfaz gráfica de usuario, lo que origina la ejecución de tareas cortas en el hilo despachador de eventos (*Event Dispatch Thread*).

No obstante, cuando el programa necesita ejecutar tareas que requieren un largo periodo de tiempo, éstas no deben ser realizadas por el despachador de eventos, sino que deben delegarse a un objeto que herede de la clase `SwingWorker`. Las subclases de `SwingWorker` deben implementar el método `doInBackground()` para realizar el cálculo en un segundo plano. A su vez, para invocar a dicho método debe invocarse el método `execute()` de la clase `SwingWorker`.

`SwingWorker` es una clase genérica, con dos parámetros de tipo. El primer parámetro de tipo especifica el tipo devuelto por el método `get()`, que es invocado por otros hilos para recuperar el objeto devuelto por `doInBackground()`. El segundo parámetro de tipo de `SwingWorker` especifica un tipo de resultados provisionales mientras la tarea de fondo sigue activa. Dichos resultados provisionales pueden publicarse llamando a `publish()`, que recibe un número variable de parámetros.

Para recopilar los resultados proporcionados por `publish()`, debe sobrescribirse el método `process()`. Este método se invoca desde el despachador de eventos; es por ello que el resultado de varias invocaciones de `publish()` a menudo es acumulado por una sola invocación de `process()`.

Una vez que el método `doInBackground()` termina, el método `done()` de `SwingWorker` es invocado automáticamente por el despachador de eventos.

2.5. Java Servlets y JSP

2.5.1. Servlets

Los *servlets* son programas Java que se ejecutan en un servidor Web y construyen páginas Web de manera dinámica. Los servlets son gestionados por un contenedor de servlets, residente también en el servidor. A diferencia de los *applets*, que se ejecutan en un cliente Web, los servlets se ejecutan exclusivamente en el servidor Web, reciben peticiones HTTP de los clientes y, como resultado, devuelven únicamente una página HTML o un archivo XML.

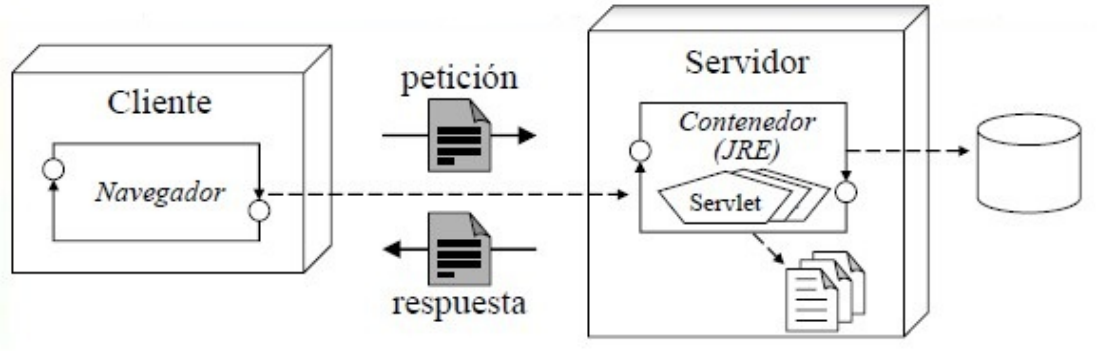


Figura 2.6: *Esquema cliente/servidor de servlets*

El proceso de ejecución de un servlet es el siguiente:

1. La petición Web llega al servidor Web.
2. La petición se traslada al contenedor de aplicaciones, en concreto a su motor del servicio Servlet/JSP, el cual posee su propia máquina virtual.
3. El motor encapsula la información de la petición en un objeto de tipo `HttpServletRequest`. Por otro lado, encapsula el flujo de respuesta en un objeto de tipo `HttpServletResponse`.
4. El motor crea, por cada petición cliente, un hilo con el que invocar al método `service()` del servlet. En función del tìpo de petición HTTP (POST o GET), `service()` llama al método correspondiente del servlet: `doPost()`, `doGet()`..., pasándole como parámetro los objetos asociados a `HttpServletRequest` y `HttpServletResponse`.
5. Cada clase del tipo servlet, tiene una única instancia, sobre la que corren los diferentes hilos, uno por cada petición HTTP.

El funcionamiento de los servlets es similar al de CGI (*Common Gateway Interface*), pero con una arquitectura diferente. Las principales ventajas de los servlets frente a otras tecnologías Web son:

- Está desarrollado en Java, por lo que es multiplaforma y orientado a objetos.
- Cada petición genera un hilo dentro de un proceso, mientras que en CGI cada petición genera un proceso. Esto permite un menor consumo de memoria, un menor retraso en las peticiones y una mayor escalabilidad.
- El servlet mantiene automáticamente su estado y recursos externos.
- Permite realizar tareas típicas de un servidor: logging, gestión de errores, sesiones...
- Los diferentes servlets pueden compartir datos entre ellos o utilizar bases de datos.
- Posee un gran número de APIs: bases de datos, red...
- Posee una comunidad muy amplia de desarrolladores.

2.5.2. Java Server Pages (JSP)

Las páginas JSP (*Java Server Pages*), son documentos HTML/XML con etiquetas especiales para programar scripts de servidor en sintaxis Java.

Los JSPs son en realidad servlets: un JSP se compila a un programa Java la primera vez que se invoca, y del programa en Java se crea una clase que se empieza a ejecutar en el servidor como un servlet. La principal diferencia entre servlets y JSPs es el enfoque de la programación:

- Un JSP es una página Web con etiquetas especiales y código Java incrustado, el cual genera dinámicamente parte del código HTML, a partir de datos incluidos en la petición HTTP u otros provenientes de otros JSPs o servlets.
- Un servlet es un programa que recibe peticiones y, a partir de ellas, genera una página HTML.

Capítulo 3

Descripción general del sistema

3.1. Planteamiento del problema

El objetivo principal del presente proyecto, como se introdujo en la sección 1.2, es el desarrollo de un sistema que permita el envío de ficheros o píldoras desde un ordenador a una serie de terminales móviles, mediante comunicación Bluetooth. Por el hecho de emplear Bluetooth en lugar de otras tecnologías de transmisión de datos, este escenario presenta una serie de peculiaridades que deben tenerse en cuenta:

- Bluetooth es una tecnología inalámbrica y permite que cualquier dispositivo Bluetooth sea capaz de descubrir a los dispositivos Bluetooth cercanos, independientemente de que realice funciones de cliente o de servidor.
- Bluetooth permite implementar ciertas características de seguridad (ver sección 2.1.6). Cada servicio Bluetooth, además, presenta un nivel de seguridad distinto.
- Los dispositivos Bluetooth se identifican mediante dos sistemas:
  - **Friendly Name**: Representa el nombre del dispositivo Bluetooth. Es de carácter alfanumérico y editable por el usuario del dispositivo. (Ej.:*Móvil de Luis*).
  - **Bluetooth Address**: Representa la dirección física y unívoca del dispositivo Bluetooth. Está escrita en hardware, por lo que no se puede modificar. Está compuesto por 6 bytes y suele representarse en sistema hexadecimal (Ej.:*001CD41B3EA5*).

- Como se comentó en la sección 2.1, la frecuencia de trabajo de Bluetooth permite que las ondas transmitidas superen obstáculos, como paredes o ventanas. Por otro lado, el alcance de comunicación entre dispositivos Bluetooth es de aproximadamente 10 metros.

Como se describe en la sección 1.3 sobre el desarrollo de las fases del proyecto, el sistema consta de dos subsistemas o aplicaciones que funcionan independientemente pero que, como veremos más adelante, comparten datos comunes entre ellas. Estas dos aplicaciones son:

- **Aplicación Bluepills**: Aplicación gráfica desarrollada íntegramente en lenguaje Java (J2SE), que realiza el envío (vía Bluetooth) de un conjunto de ficheros desde un ordenador a los terminales móviles de los usuarios registrados. Constituye el eje principal del sistema. Es el resultado del desarrollo de las fases 1, 3 y 5 del proyecto.
- **Aplicación Web Bluepills**: Aplicación Web desarrollada en lenguaje Java (J2EE), que permite registrar los datos Bluetooth de los terminales móviles de los usuarios. Corresponde al desarrollo de las fases 2 y 4 del proyecto.

3.2. Estructura general del sistema

El sistema a desarrollar en el presente proyecto plantea un escenario compuesto por una serie de entidades que se relacionan siguiendo el esquema de la figura 3.1.

Dicha figura muestra la relación entre el ordenador que envía los ficheros y los dispositivos móviles sobre los que se realiza el envío de ficheros. En lo sucesivo, denominaremos "*equipo Bluetooth*" a la máquina en la que corre la aplicación Bluepills y "*terminales Bluetooth*" a los terminales móviles equipados con Bluetooth. A su vez, al usuario del equipo Bluetooth le denominaremos "*usuario del equipo*", y a los usuarios de los terminales móviles Bluetooth, "*usuarios finales*".

Como puede observarse en la figura 3.1, la transmisión de datos se realiza en un único sentido, desde el equipo Bluetooth a los terminales Bluetooth. Los papeles que desempeñan tanto el equipo como los terminales Bluetooth son los siguientes:

- **Equipo Bluetooth**: En él reside la aplicación Bluepills en ejecución. Localiza a los terminales Bluetooth cercanos y realiza el envío de ficheros. Debe disponer de un dispositivo hardware Bluetooth.

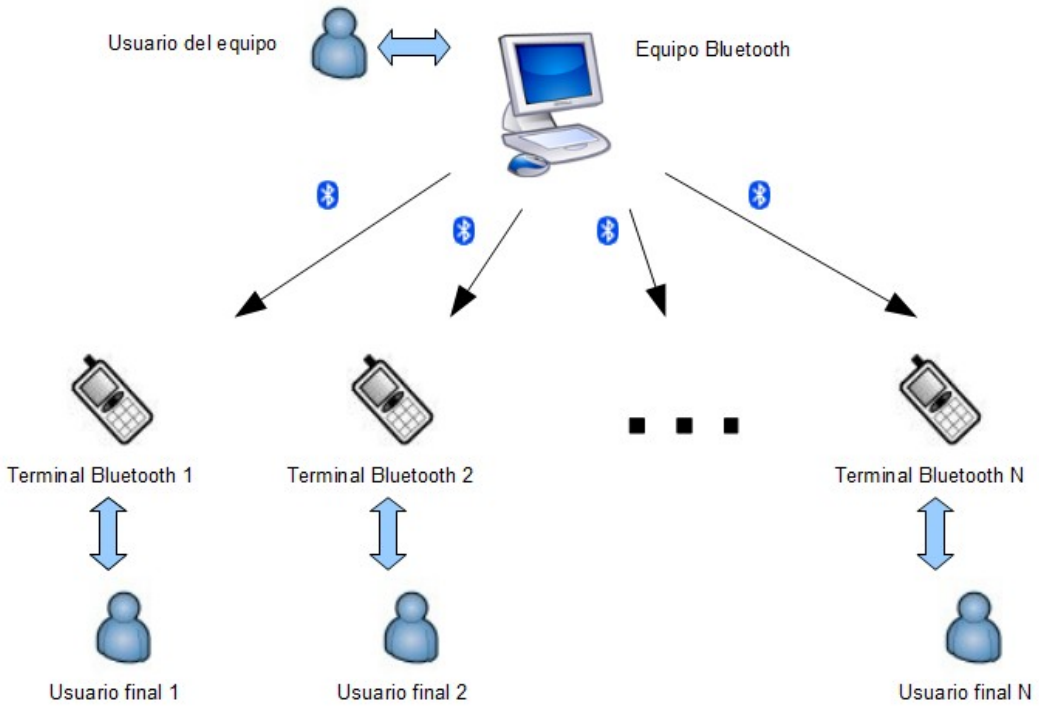


Figura 3.1: *Relación entre el equipo y los terminales Bluetooth*

- **Terminal Bluetooth**: Permite ser localizado por el equipo Bluetooth y recibe el envío de ficheros. No requiere la instalación ni ejecución de ninguna aplicación.

Por otro lado, las funciones que desempeñan los distintos usuarios del sistema son las siguientes:

- **Usuario del equipo**: Interactúa con la aplicación Bluepills, seleccionando los ficheros que desea enviar e iniciando el proceso de envío. A través de la interfaz gráfica ofrecida por la aplicación, puede acceder a cierta información relativa a los dispositivos Bluetooth circundantes.
- **Usuario final**: Interactúa exclusivamente con su terminal móvil correspondiente. Realiza las siguientes tareas:
  1. Activa el sistema Bluetooth en su terminal, de manera que pueda ser detectado por otros dispositivos Bluetooth.
  2. Acepta solicitudes de envío de ficheros por parte del equipo Bluetooth.

Como se comentó en la sección 3.1, el sistema Bluepills consta también de una aplicación Web que permite el registro de los usuarios al sistema. Esta aplicación Web reside en un servidor Web (que puede ser remoto o local), y a él acceden los usuarios a través de un navegador Web. A esta aplicación Web también pueden acceder otro tipo de usuarios, con perfil de administrador. Las funciones de ambos, usuario a registrar y administrador, son las siguientes:

- **Nuevo Usuario**: Registra sus datos en la aplicación Web, introduciendo su nombre de usuario y el nombre de su dispositivo Bluetooth (Friendly Name).
- **Administrador**: Accede a la aplicación Web para consultar, modificar o borrar datos de los usuarios registrados.

3.3. Funcionamiento general del sistema

En la figura 3.2 se muestra la relación entre la aplicación Bluepills y la aplicación Web. El funcionamiento de todo el sistema, aplicación Bluepills y aplicación Web, consta de varias fases:

1. Los usuarios finales, utilizando un navegador Web, acceden a la aplicación Web y registran sus datos. Dichos datos quedan almacenados en el servidor Web en forma de ficheros.

2. El usuario del equipo Bluetooth importa los datos de registro de usuario, almacenados en el servidor Web, a través de la aplicación Bluepills. La aplicación Bluepills puede residir en la misma máquina que ejerce de servidor Web o en otra.

3. El usuario del equipo Bluetooth, utilizando la aplicación Bluepills, selecciona un conjunto de ficheros en su equipo. A continuación, la aplicación Bluepills envía los ficheros seleccionados, exclusivamente, a los terminales Bluetooth de los usuarios finales registrados.

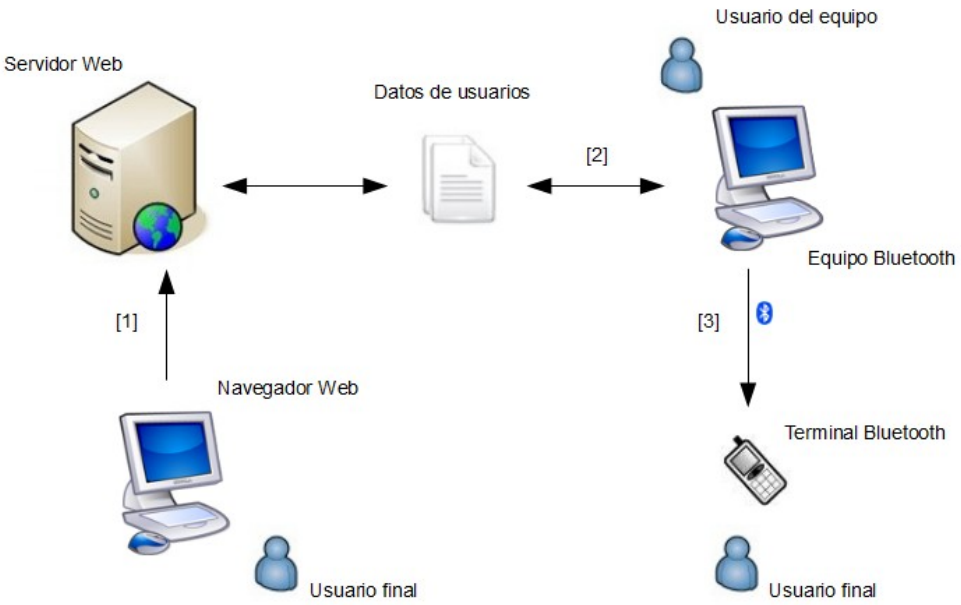


Figura 3.2: *Esquema de funcionamiento del sistema Bluepills*

3.4. Modelo de ficheros

Como se ha comentado anteriormente, en principio la aplicación Bluepills no realiza el envío de ficheros a todos los dispositivos Bluetooth circundantes, sino únicamente a los terminales móviles de los usuarios registrados. La información relativa a dichos usuarios se almacena en varios ficheros en formato *.csv*.

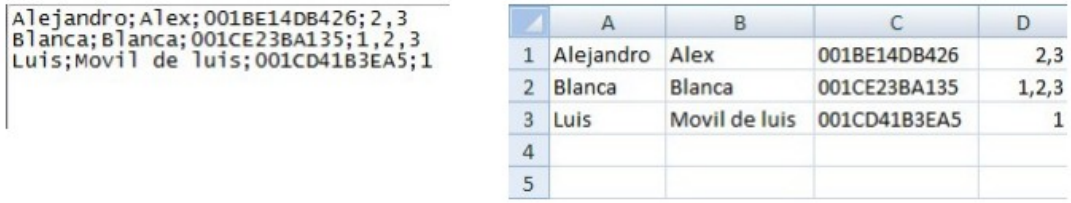
Los ficheros CSV (*Comma-Separated Values*) son ficheros de texto sencillos que permiten representar datos en forma de tabla: las columnas se separan por un carácter delimitador y las filas por saltos de línea. El carácter delimitador depende de la configuración regional del sistema operativo; en el caso de España, el delimitador por defecto es el punto y coma (';'), para no confundirlo con la coma decimal empleada en datos numéricos. Existen numerosos programas capaces de interpretar estos ficheros (por ejemplo, Microsoft Excel) y proporcionar un formato visual en forma de tabla.

3.4.1. Fichero de usuarios

El fichero CSV de usuarios registrados, utilizado por la aplicación Bluepills, consta de los siguientes campos, separados por punto y coma:

- **Nombre del usuario**: Describe unívocamente al usuario del sistema.
- **Friendly Name**: Nombre del dispositivo Bluetooth del terminal del usuario (ver sección 3.1).
- **Bluetooth Address**: Dirección física del dispositivo Bluetooth del terminal del usuario (ver sección 3.1).
- **Grupos**: Lista con los identificadores de los grupos de usuarios en los que está inscrito el usuario. Cada identificador se separa con el carácter coma (',').

En la figura 3.3 se muestra un ejemplo de un fichero de usuarios y su interpretación en un lector de ficheros CSV.



```
Alejandro;Alex;001BE14DB426;2,3
Blanca;Blanca;001CE23BA135;1,2,3
Luis;Movil de luis;001CD41B3EA5;1
```

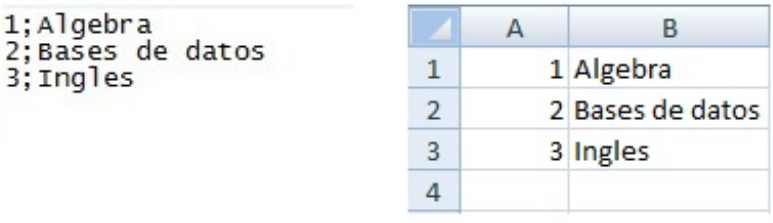
	A	B	C	D
1	Alejandro	Alex	001BE14DB426	2,3
2	Blanca	Blanca	001CE23BA135	1,2,3
3	Luis	Movil de luis	001CD41B3EA5	1
4				
5				

Figura 3.3: *Ejemplo de fichero de usuarios*

3.4.2. Fichero de grupos

Además del fichero de usuarios, el sistema Bluepills utiliza otro fichero CSV, para gestionar grupos de usuarios. Este fichero consta de los siguientes campos, separados por punto y coma:

- **Identificador de grupo**: Describe unívocamente a un grupo, mediante un identificador numérico.
- **Nombre de grupo**: Nombre del grupo de usuarios.



```
1;Algebra
2;Bases de datos
3;Ingles
```

	A	B
1	1	Algebra
2	2	Bases de datos
3	3	Ingles
4		

Figura 3.4: *Ejemplo de fichero de grupos*

Alternativamente al envío de ficheros a usuarios registrados, la aplicación Bluepills permite filtrar a los usuarios por grupos. Cada usuario registrado puede pertenecer a uno, a varios o a ningún grupo. Seleccionando la opción correspondiente, la aplicación puede realizar el envío de ficheros únicamente a los miembros de un determinado grupo, en lugar de a todos los usuarios registrados. En un entorno docente, por ejemplo, esta clasificación en grupos puede emplearse para separar alumnos por asignaturas.

3.4.3. Fichero de sesiones

Cada ejecución de la aplicación Bluepills para el envío de ficheros se enmarca en un contexto que denominamos "sesión". Al término de cada sesión, es posible almacenar los datos correspondientes a la misma en un fichero de sesiones. Posteriormente, si el usuario del equipo realiza un nuevo envío de ficheros, puede almacenar los datos de la nueva sesión en un nuevo fichero de sesiones o en un fichero de sesiones existente. Cada fichero de sesiones almacena los siguientes datos, en formato CSV:

- **Nombre del usuario**: Describe unívocamente al usuario del sistema. Coincide con el nombre de usuario del fichero de usuarios.
- **Fecha**: Fecha de la sesión.
- **Ficheros**: Lista de ficheros enviados durante la sesión. Cada nombre de fichero se separa con el carácter coma (','').
- **Fecha**: ...
- **Ficheros**: ...
- **Fecha**: ...
- **Ficheros**: ...

```
Alejandro;20/12/2010;documento.txt
Blanca
Luis;11/12/2010;apuntes.pdf,imagen.bmp;20/12/2010;documento.txt
```

	A	B	C	D	E
1	Alejandro	20/12/2010	documento.txt		
2	Blanca				
3	Luis	11/12/2010	apuntes.pdf,imagen.bmp	20/12/2010	documento.txt
4					
5					

Figura 3.5: *Ejemplo de fichero de sesiones*

En la figura 3.5 se muestra un ejemplo de un fichero de sesiones. En el ejemplo, el usuario "*Alejandro*" recibió el fichero "*documento.txt*" durante la sesión del día 20 de diciembre de 2010. El usuario "*Luis*", por su parte, participó en dos sesiones: la sesión del día 11 de diciembre, en la que recibió dos ficheros, y la sesión del día 20 de diciembre.

Capítulo 4

Desarrollo de la aplicación Bluepills

4.1. Estructura

La aplicación Bluepills está desarrollada íntegramente en lenguaje Java y consta de tres módulos, representados en la figura 4.1.

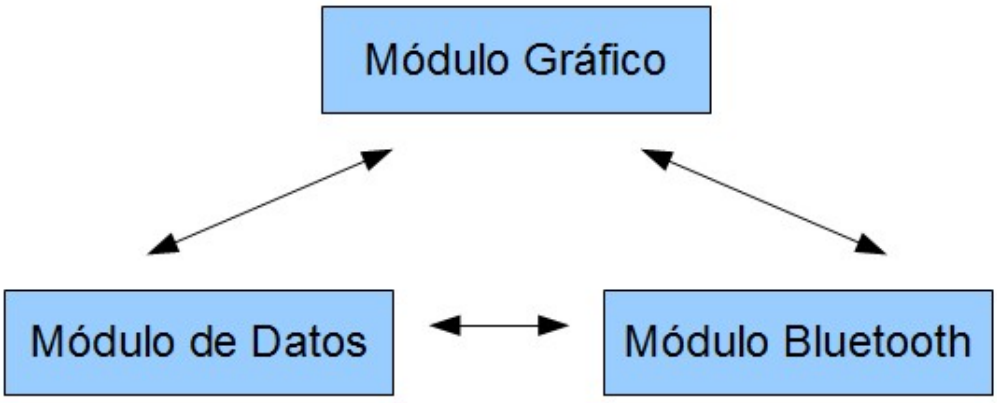


Figura 4.1: *Módulos de la aplicación Bluepills*

Cada uno de estos módulos está compuesto por una serie de clases Java con una funcionalidad determinada pero, no obstante, todos ellos se relacionan entre sí, ya sea para acceder a datos compartidos o para invocar alguno de sus métodos.

A grandes rasgos, la funcionalidad de cada uno de los módulos Bluepills es la siguiente:

- **Módulo Gráfico**: Presenta una interfaz gráfica entre la aplicación y el usuario del equipo. Permite seleccionar los ficheros a enviar en cada sesión, acceder a la información relativa a los terminales Bluetooth circundantes e iniciar el envío de ficheros.
- **Módulo Bluetooth**: Gestiona todo lo concerniente a las comunicaciones Bluetooth (búsqueda de dispositivos, establecimiento de conexiones, envío de ficheros...) utilizando los APIs de la librería BlueCove (ver sección 2.3.2).
- **Módulo de Datos**: Gestiona la lectura y escritura de todos los ficheros de datos (ficheros de usuarios, de grupos y de sesiones). También gestiona las estructuras de datos asociadas a dichos ficheros, que son compartidas por los tres módulos.

En la figura 4.2 se muestra la estructura, en forma de árbol, de los distintos paquetes y clases java que componen la aplicación Bluepills. La relación de los módulos de la aplicación con dichos paquetes es la siguiente:

- Módulo Gráfico: paquete `GUI`.
- Módulo Bluetooth: paquete `BTI`.
- Módulo de Datos: paquete `users`.

A continuación, en las siguientes secciones, se analiza en detalle la estructura y el funcionamiento de cada uno de los tres módulos.

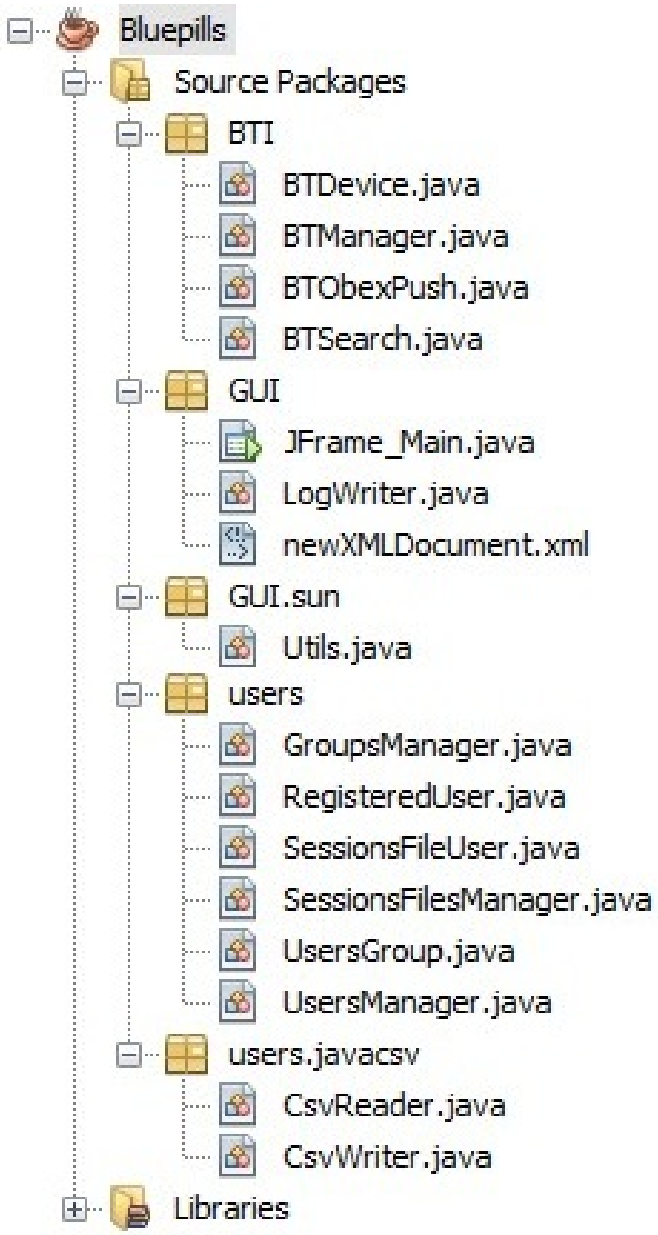


Figura 4.2: *Diagrama de clases de la aplicación Bluepills*

4.2. Módulo Bluetooth

Aunque el eje principal de la aplicación Bluepills es el Módulo Gráfico, ya que de él parte el punto de inicio del programa, el núcleo fundamental del sistema reside en el Módulo Bluetooth, pues es el encargado de realizar todo el proceso de comunicación con los terminales Bluetooth. Este módulo está compuesto por un conjunto de clases Java que, a su vez, utilizan el API `javax.bluetooth` y `javax.obex`, pertenecientes a la librería BlueCove, para poder interactuar con el dispositivo hardware Bluetooth del equipo. Las principales funciones de este módulo son:

- Realizar búsquedas de dispositivos Bluetooth remotos.
- Obtener información de los servicios ofrecidos por los dispositivos Bluetooth remotos.
- Obtener información sobre las características de los dispositivos Bluetooth remotos, como su dirección física o su nombre.
- Establecer comunicaciones OBEX Bluetooth con los dispositivos remotos para la transmisión de ficheros.

4.2.1. El modelo cliente-servidor Bluetooth

Al igual que en muchos otros sistemas de comunicaciones, el modelo de comunicaciones Bluetooth no es simétrico. Por ello, antes de desarrollar en Java cualquier código relacionado con las comunicaciones Bluetooth, es necesario entender el papel y las funciones que desempeñan cada uno de los terminales situados en ambos extremos de la comunicación.

Cuando se establece un enlace Bluetooth entre dos dispositivos, uno de los dispositivos se define como "maestro" y otro como "esclavo" (ver sección 2.1.2). El estándar Bluetooth establece que el dispositivo que inicia la conexión asume el papel de maestro, mientras que el dispositivo que acepta la conexión ejerce el papel de esclavo. El rol de maestro no otorga ningún privilegio sobre el de esclavo, pero implica que será el dispositivo maestro el encargado de calcular la secuencia de saltos en frecuencia, así como de controlar la sincronización entre ambos dispositivos. Adicionalmente, algunos dispositivos Bluetooth son capaces de soportar la conmutación entre ambos papeles, permitiendo que en cualquier momento el dispositivo maestro se convierta en esclavo y viceversa. En conexiones entre dos dispositivos no es relevante en realidad quién ejerce de maestro y quién

de esclavo. Sin embargo, en conexiones entre múltiples dispositivos, sólo pueden existir un dispositivo maestro y siete esclavos funcionando simultáneamente, y hasta 255 esclavos en estado inactivo.

En la figura 4.3 se muestra la relación, a distintos niveles, entre un dispositivo Bluetooth cliente y un dispositivo Bluetooth servidor. En un principio, según el esquema de la relación entre el equipo y los terminales Bluetooth (ver figura 3.1), parece lógico pensar que, en nuestro modelo de comunicación, es el equipo Bluetooth el que ejerce de servidor y que son los terminales Bluetooth los que ejercen de clientes. No obstante, en realidad el modelo no funciona exactamente así, como veremos a continuación. El equipo Bluetooth no transmite simultáneamente los ficheros a todos los terminales Bluetooth sino que, tras realizar una búsqueda de dispositivos y servicios, establece una conexión Bluetooth independiente con cada uno de ellos. El modo en que se realizan dichas conexiones determina los roles cliente/servidor de cada uno de los dispositivos, a distintos niveles:

- **Nivel de aplicación**: A nivel de aplicación, el envío de ficheros se realiza utilizando la operación PUT del protocolo OBEX (ver sección 2.2), la cual determina que es el cliente el que realiza el envío o carga de objetos, mientras que el servidor es el receptor de los mismos.
- **Nivel JSR-82**: A este nivel, es el terminal Bluetooth el que ofrece el servicio OBEX Object Push, y el equipo Bluetooth el que realiza la búsqueda del servicio en el terminal. Una vez descubierto el servicio, es el cliente (equipo Bluetooth) el que inicia la conexión con el servidor (terminal Bluetooth).
- **Nivel Bluetooth**: Como comentamos en el párrafo anterior, todo dispositivo que inicia la comunicación se convierte en maestro y, por ende, el otro dispositivo se convierte en esclavo. No obstante, si se especifica en los parámetros del servicio y los dispositivos lo permiten, durante la comunicación estos papeles pueden intercambiarse.

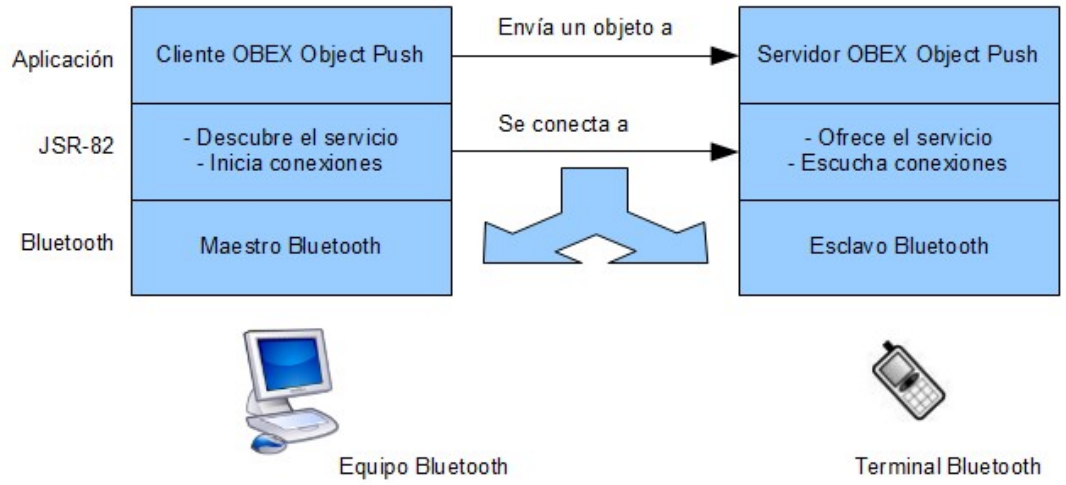


Figura 4.3: *Modelo cliente-servidor en Bluepills*

4.2.2. Comunicación cliente-servidor OBEX

Como se comentó en la sección anterior, en la comunicación OBEX entre el equipo y un terminal Bluetooth, es éste último el que ejerce la función de servidor. Además, todos los terminales móviles del mercado equipados con la tecnología Bluetooth implementan el servicio OBEX Object Push (para poder enviar y recibir objetos entre dispositivos), por lo que no es necesario el desarrollo de código adicional en la parte del servidor.

El perfil GOEP de la especificación Bluetooth [1] determina los roles definidos para clientes y servidores OBEX:

- **Servidor OBEX**: proporciona un servidor de intercambio de objetos, hacia o desde el cual los objetos pueden ser enviados (push) o solicitados (pull).
- **Cliente OBEX**: Envía o solicita objetos hacia o desde el servidor.

Para permitir que el dispositivo cliente pueda encontrar al dispositivo servidor y establecer conexiones con él, es necesario que este último se encuentre en Modo Visible y en Modo Conectable. Para ello, el usuario final debe activar el dispositivo Bluetooth en su terminal móvil y configurarlo para tal propósito.

El perfil GOEP especifica que si la autorización o la autenticación OBEX están soportadas, y la implementan tanto el servidor como el cliente, ésta debe inicializarse antes de establecer la sesión OBEX entre ambos dispositivos. Para

el caso particular del servicio OBEX Object Push, éste utiliza un nivel de seguridad Modo 2 (ver sección 2.1.6), por lo que el único mecanismo de seguridad implementado es el de autorización. En este caso, antes de que el cliente inicie una sesión OBEX con el dispositivo servidor, éste es notificado del intento de conexión, mostrándole por pantalla al usuario el nombre del dispositivo cliente que intenta conectarse con él. En caso de que el usuario final la autorice (pulsando la opción correspondiente en su terminal), la sesión OBEX puede iniciarse. En caso contrario, la conexión se rechaza.

En la siguiente figura se muestra el diagrama de intercambio de mensajes entre cliente y servidor, durante una sesión OBEX en Bluepills, una vez que el proceso de autorización ha concluido con éxito.

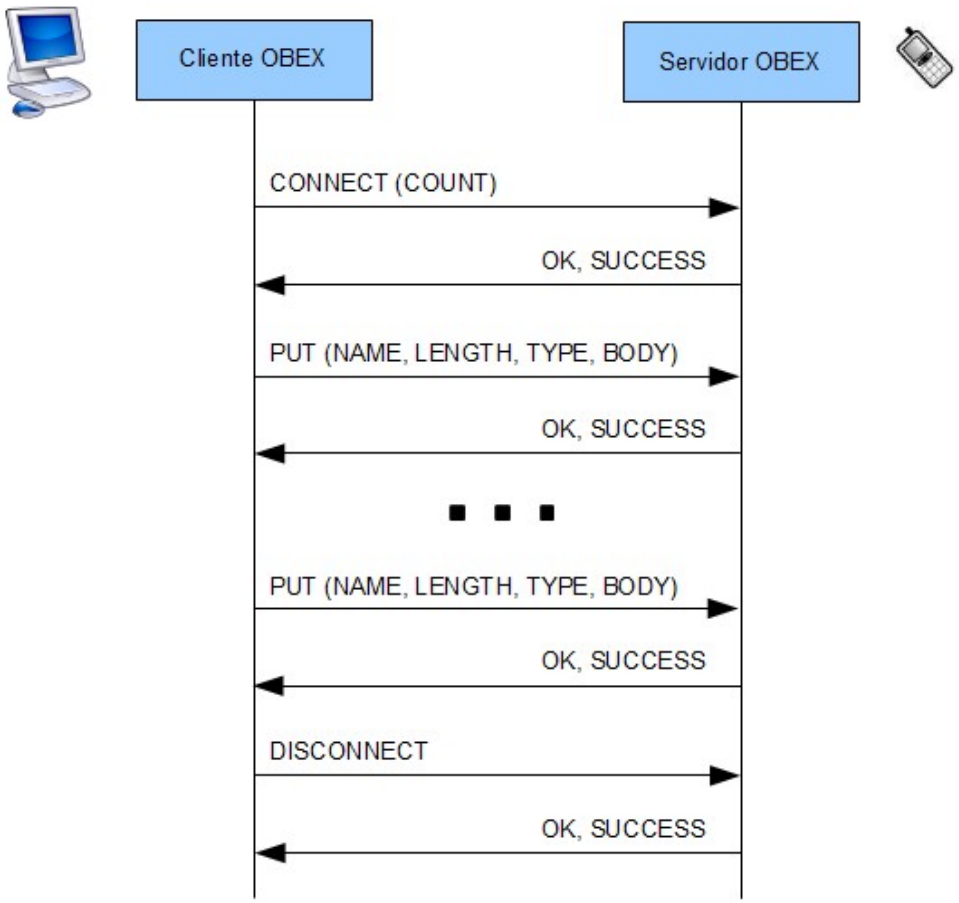


Figura 4.4: *Diagrama de comunicación OBEX en Bluepills*

Como puede observarse en la figura 4.4, el cliente inicia cada operación OBEX con un mensaje de petición y, a continuación, espera una respuesta del servidor. Una sesión OBEX comienza con una petición CONNECT del cliente al servidor. A su vez, la sesión finaliza con una petición DISCONNECT. Entre las peticiones CONNECT y DISCONNECT, el cliente puede enviar un número ilimitado de peticiones PUT, GET o SETPATH al servidor. En el caso de la aplicación Bluepills, tras la petición CONNECT sólo se envían peticiones PUT.

Cada mensaje de petición y respuesta anterior incluye una o más cabeceras OBEX. Como se describió en la sección 2.3.4, estas cabeceras puede ser de distintos tipos: NAME, LENGTH, TYPE, COUNT, DESCRIPTION... En el caso particular de la operación CONNECT en la implementación Bluepills, ésta incluye una cabecera de tipo COUNT, en la que se indica el número de objetos o ficheros a enviar. Cuando el dispositivo móvil servidor recibe la petición, procesa las cabeceras recibidas y determina si acepta o rechaza la conexión. Si el servidor acepta la conexión, responde con un código de respuesta de tipo "OK, SUCCESS". Si por el contrario el servidor rechaza la conexión, éste responde con un código de respuesta HTTP que especifica el motivo del rechazo.

Una vez que la conexión se ha establecido satisfactoriamente, el cliente emite una petición PUT al servidor, con el fin de transmitir el primero de los ficheros. En este caso, el mensaje OBEX incluye cuatro cabeceras:

- NAME: nombre del fichero a enviar.
- TYPE: tipo MIME del objeto a enviar; al tratarse de un fichero, el tipo es "object".
- LENGTH: tamaño del fichero a enviar, en número de bytes.
- BODY: especifica que dentro del mensaje OBEX van incluidos bytes de datos, correspondientes a un objeto.

Una vez que el objeto ha sido transmitido correctamente al servidor, éste responde de nuevo con un código de respuesta "OK, SUCCESS". En caso contrario, responde con el código de respuesta HTTP pertinente.

A partir de este punto, continúan enviándose tantas peticiones PUT como número de ficheros haya pendientes de enviar. Para finalizar la sesión OBEX, el cliente debe enviar una petición DISCONNECT. Generalmente, la petición DISCONNECT no requiere el uso de cabeceras adicionales, aunque éstas podrían incluirse. Cuando recibe una petición DISCONNECT, el servidor libera los recursos reservados y envía una respuesta "OK, SUCCESS" al cliente. Cuando el cliente recibe dicha respuesta, la sesión OBEX termina.

En el transcurso de la sesión OBEX, si el cliente recibe un código de respuesta HTTP distinto a "OK, SUCCESS" o la transmisión de alguno de los objetos se interrumpe por cualquier motivo, el cliente envía una petición DISCONNECT al servidor, con el fin de cerrar la sesión y evitar que éste se bloquee indefinidamente, esperando una nueva petición OBEX del cliente.

4.2.3. Sesión de envío de ficheros

Una vez expuesto el proceso de comunicación cliente-servidor en una sesión Bluepills de transferencia de ficheros, a continuación se describen los pasos necesarios para poder llevarlo a cabo, desde un punto de vista esquemático. En la siguiente sección, no obstante, se describen las clases y métodos Java implicados en dicha tarea.

A nivel de los módulos que componen la aplicación Bluepills, el Módulo Bluetooth es invocado por el Módulo Gráfico cada vez que se realiza un nuevo envío de ficheros a los dispositivos Bluetooth remotos. A través de la interfaz gráfica ofrecida por el Módulo Gráfico, el usuario del equipo puede iniciar en cualquier momento una nueva sesión de envío de ficheros. Cuando esto sucede, el Módulo Bluetooth (invocado por el Módulo Gráfico) ejecuta una secuencia de pasos, representada en el diagrama de flujo de la figura 4.5 y resumida en los siguientes puntos:

1. El usuario de la aplicación inicia la sesión.
2. El Módulo Bluetooth inicia una búsqueda de dispositivos Bluetooth remotos y permanece a la espera.
3. Una vez que la búsqueda de dispositivos Bluetooth remotos ha concluido, contabiliza el número de dispositivos remotos encontrados.
4. Accede a la información de los usuarios registrados, a través del fichero de usuarios.
5. Compara la información de los dispositivos encontrados con la de de los usuarios registrados; si alguno de los datos (Friendly Name o Bluetooth Address) ha cambiado, actualiza el fichero de usuarios.
6. Para el dispositivo 'n' encontrado, si éste se encuentra registrado, realiza una búsqueda del servicio OBEX Object Push.
7. Si se encuentra el servicio, crea un hilo independiente, que se encargará de realizar la comunicación OBEX para el envío de ficheros al dispositivo.

8. Si hay más dispositivos remotos encontrados, vuelve al paso 4. En caso contrario, la sesión de envío de ficheros termina. No obstante, como se describe en la sección 4.3.1, el Módulo Gráfico invoca de nuevo este proceso, cíclicamente, hasta que el usuario solicita su detención.

El servicio OBEX Object Push (ver sección 2.2.2) proporciona un mecanismo de alto nivel para la transmisión de ficheros desde el equipo Bluetooth hacia los terminales Bluetooth. No obstante, como puede deducirse en el diagrama de flujo de la figura 4.5, el envío de ficheros a cada uno de los terminales no es simultáneo sino que, tras verificar que éstos implementan el servicio, la aplicación abre una conexión distinta con cada uno de ellos. Cada una de estas conexiones se delega en un nuevo hilo de ejecución independiente, evitando así el envío secuencial (y bloqueante) de los ficheros.

Cada hilo de comunicación OBEX ejecuta a su vez una secuencia de pasos, los cuales se describen gráficamente en la figura 4.6. Estos pasos son:

1. El hilo de comunicación OBEX obtiene la URL (*Uniform Resource Locator*) del servicio, a partir de los datos obtenidos en la búsqueda del servicio (ver figura 4.5).

2. Intenta establecer una conexión con el dispositivo remoto, utilizando la URL obtenida anteriormente.

3. El terminal del usuario, al recibir el intento de conexión Bluetooth, solicita permiso al usuario. Si el usuario rechaza la conexión, la ejecución del hilo termina.

4. Una vez que el usuario ha aceptado la conexión, envía un mensaje OBEX CONNECT a su dispositivo remoto, y espera una respuesta a dicho mensaje.

5. Si la respuesta del mensaje OBEX CONNECT es correcta, continúa; en caso contrario, envía un mensaje OBEX DISCONNECT al dispositivo remoto y la ejecución termina.

6. Envía un mensaje OBEX PUT al dispositivo remoto, con uno de los ficheros como cuerpo del mensaje. Si durante la transmisión del fichero se produce algún error, envía un mensaje OBEX DISCONNECT al dispositivo remoto y la ejecución termina.

7. Si hay más archivos pendientes de enviar, vuelve al paso 6. En caso contrario, la ejecución termina.

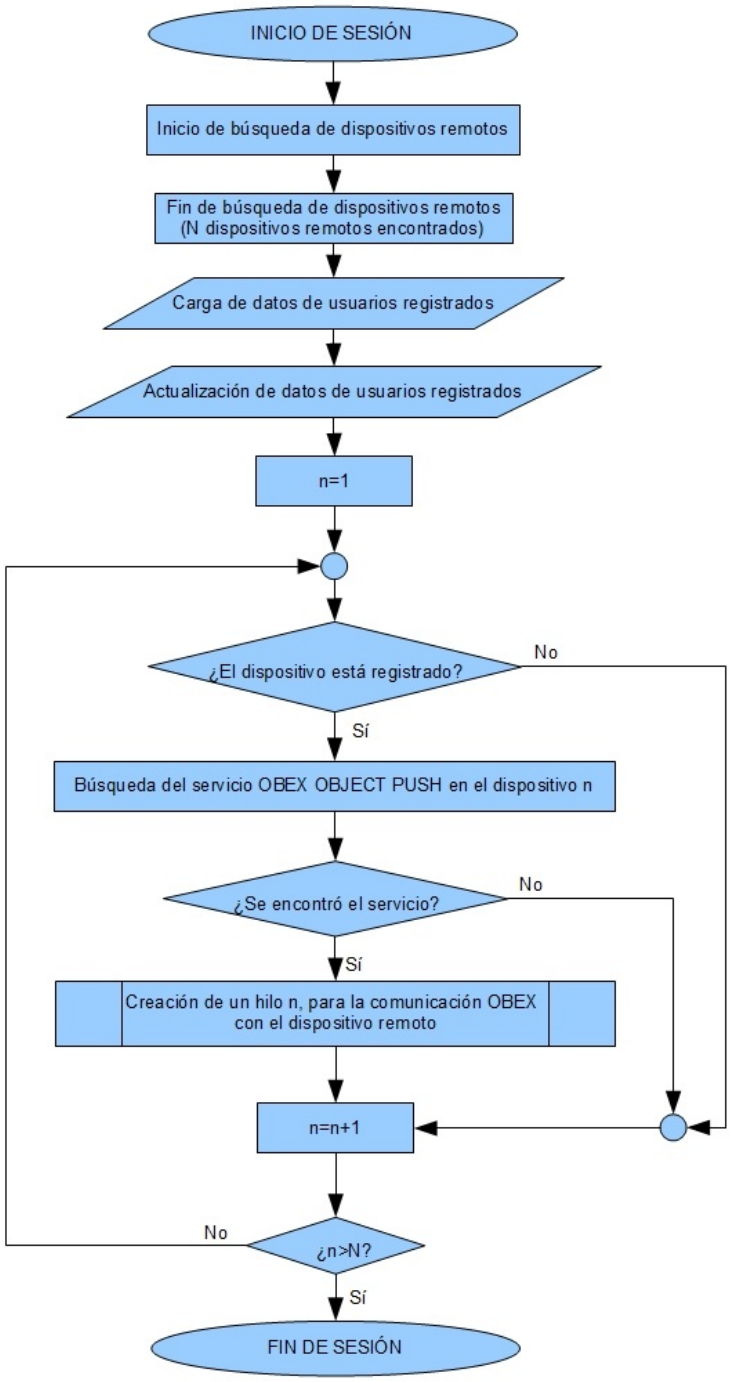


Figura 4.5: *Diagrama de flujo: Sesión de envío de ficheros*

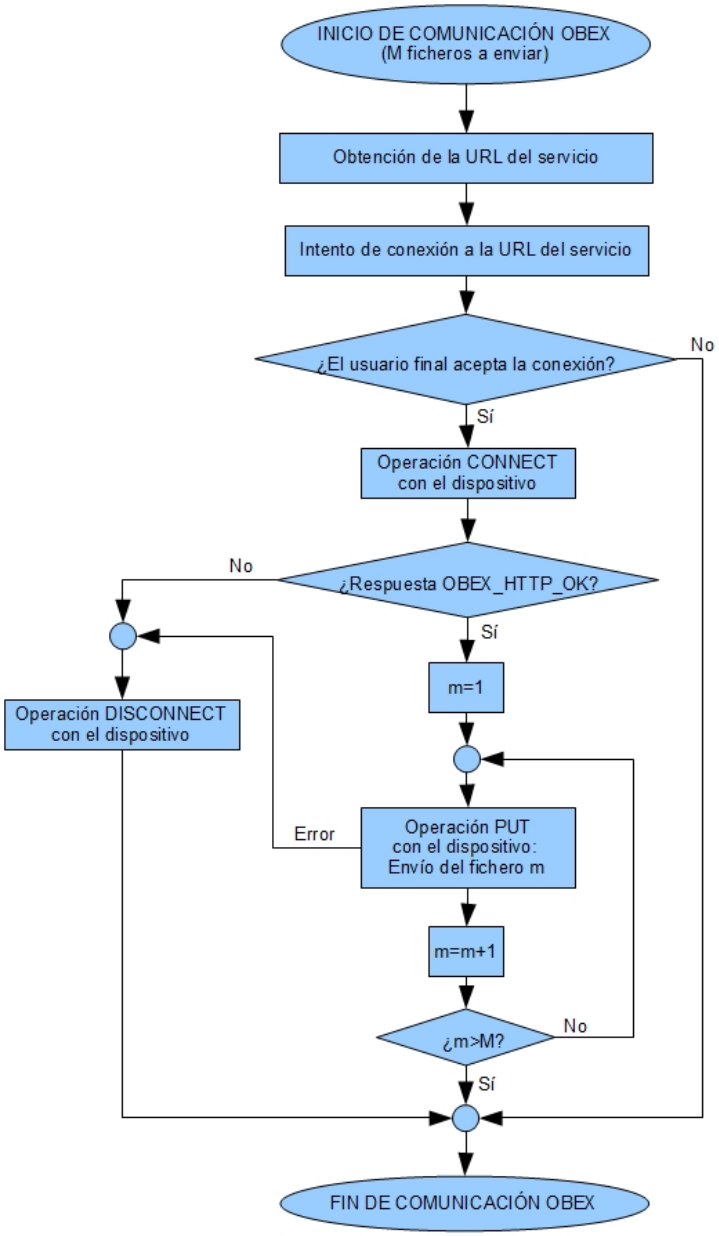


Figura 4.6: *Diagrama de flujo: Comunicación OBEX*

4.2.4. Estructura de clases del Módulo Bluetooth

Como se ha apuntado anteriormente, la función principal del Módulo Bluetooth es la de implementar el proceso de comunicación entre el equipo y los terminales Bluetooth. Para ello, las clases Java de este módulo utilizan las librerías JSR-82 proporcionadas por BlueCove para poder interactuar con los protocolos inferiores del dispositivo hardware Bluetooth del equipo. Dentro de la aplicación Bluepills, dichas clases están contenidas en el paquete Java `BTI` (*Bluetooth Interface*). La funcionalidad de cada una de estas clases es la siguiente:

`BTManager`

Constituye el eje principal del Módulo Bluetooth. Ejerce de nexo entre el Módulo Gráfico y los otros dos módulos, invocando a los métodos de éstos. Utiliza estructuras de datos compartidas con el Módulo de Datos.

`BTSearch`

Clase que implementa la búsqueda de dispositivos y servicios Bluetooth remotos. Posee, entre otros, los siguientes métodos:

- `searchDevices()`: Solicita una búsqueda de los dispositivos Bluetooth circundantes. Tras esto, el hilo de ejecución permanece a la espera de que la búsqueda de dispositivos concluya.
- `deviceDiscovered()`: Cada vez que un dispositivo Bluetooth remoto es encontrado, este método es invocado por el escuchador de eventos. A través de este objeto puede obtenerse información relativa al dispositivo remoto encontrado.
- `inquiryCompleted()`: Una vez que la búsqueda de dispositivos ha concluido, el escuchador de eventos invoca a este método, indicando el motivo de la finalización: búsqueda completada, búsqueda cancelada o error en la búsqueda. Además, reactiva al hilo de ejecución que permanecía a la espera tras invocar a `searchDevices()`.
- `searchService()`: Análogamente a la búsquedas de dispositivos remotos, insta al protocolo SDP a que realice una búsqueda de servicios en un determinado dispositivo remoto. Este método recibe como parámetros un array con los UUID (*Universal Unique IDentifier*) de los servicios buscados y un array con los atributos que se desean obtener de dichos servicios (tipo de

servicio, nombre...). Cada servicio estandarizado posee su propio UUID, y existe un rango de numeración disponible para cualquier nueva aplicación que se cree. En el caso del servicio OBEX Object Push, el UUID asociado es el 0x1105 (en sistema hexadecimal). Para el caso de los atributos de servicio, éstos poseen también su propia numeración. Por ejemplo, el atributo de servicio que describe el nombre del servicio es el 0x0100. Al igual que en la búsqueda de dispositivos, tras invocar una búsqueda de servicios el hilo de ejecución permanece a la espera.

- `servicesDiscovered()`: Una vez encontrados los servicios buscados, el escuchador de eventos invoca a este método, el cual recibe un array de objetos `ServiceRecord`, con una instancia por cada servicio encontrado. A través del método `getConnectionURL()` de la clase `ServiceRecord` se obtiene la URL del servicio, con una serie de parámetros de conexión. En el caso del servicio OBEX Object Push, la URL obtenida mostraría el siguiente aspecto:

  ```
  btgoep://001CD41B3EA5:9;authenticate=false;encrypt=false;
  master=false
  ```

  En este ejemplo, vemos que el perfil utilizado es GOEP, "*001CD41B3EA5*" es la dirección física del dispositivo Bluetooth y "*9*" el valor del puerto asociado al servicio. El servicio OBEX Object Push, por defecto, no implementa ni autenticación ni cifrado. El último parámetro indica que el cliente no impondrá su rol de maestro durante el tiempo que dure la conexión, sino que ambos papeles podrán ser intercambiados.

- `serviceSearchCompleted()`: Cuando la búsqueda de servicios finaliza, el escuchador de eventos invoca a este método, indicando el motivo de la finalización: búsqueda finalizada con normalidad, búsqueda cancelada, error en la búsqueda, no se encontró el dispositivo o no existe información del servicio solicitado. Al igual que en la búsqueda de dispositivos, reactiva al hilo de ejecución que permanecía a la espera tras invocar a `searchServices()`.

`BTObexPush`

Realiza todo el proceso de comunicación OBEX, el cual permite la transmisión de ficheros a través del servicio OBEX Object Push. Como se describe en la figura 4.5, por cada dispositivo remoto sobre el que se realiza el envío de ficheros, la instancia de la clase `BTManager` crea un hilo de ejecución independiente asociado a una instancia de esta clase. El constructor de esta clase recibe la URL del servicio OBEX Object Push obtenido previamente en la búsqueda de

servicios asociados al dispositivo, así como la ruta del directorio y el nombre de los ficheros a enviar. Los métodos principales de esta clase son:

- `sendFiles()`: Implementa el establecimiento de una sesión OBEX de la aplicación cliente con el servidor OBEX Object Push. Una vez establecida la sesión mediante la operación CONNECT de OBEX, realiza una llamada al método `putFile()` por cada fichero a enviar. Una vez realizadas todas las operaciones PUT necesarias, cierra la sesión OBEX mediante la operación DISCONNECT.
- `putFile()`: Implementa la operación PUT de OBEX, creando un canal de flujo de bytes que permite la transmisión del fichero desde el equipo cliente hasta el dispositivo móvil servidor. En caso de producirse un error en la transmisión del fichero, este método lanza una excepción y envía un mensaje DISCONNECT al servidor, con el fin de cerrar la sesión. De esta manera, se evita que el terminal Bluetooth quede bloqueado a la espera de respuesta por parte del equipo Bluetooth.

`BTDevice`

Estructura de datos que representa a un dispositivo remoto encontrado. Posee una serie de atributos, muchos de los cuales son mostrados en la tabla de dispositivos encontrados (ver sección 4.3). Algunos de estos atributos, como el nombre de usuario y los grupos de usuarios a los que pertenece, se obtienen de los datos del usuario asociado a este dispositivo.

4.3. Módulo Gráfico

El Módulo Gráfico representa el núcleo principal de la aplicación Bluepills. Además de ser el punto de partida de ejecución del programa, presenta una interfaz gráfica al usuario del equipo Bluetooth que permite:

- Mostrar la información relativa a los terminales Bluetooth encontrados tras realizar búsquedas de dispositivos.
- Crear sesiones de envío de ficheros a dispositivos Bluetooth remotos de usuarios registrados, mostrando un seguimiento del mismo en tiempo real.
- Seleccionar el directorio donde se hallan los ficheros a enviar.
- Llevar un seguimiento de varias sesiones realizadas en días distintos.
- Realizar el reenvío de ficheros a cualquier dispositivo Bluetooth remoto.

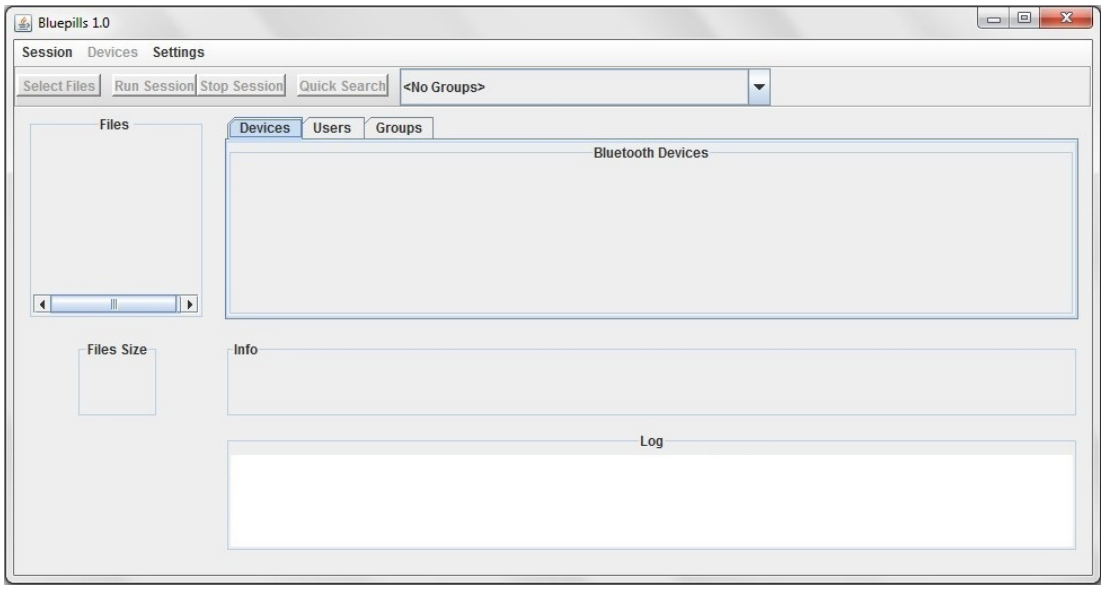


Figura 4.7: *Interfaz Gráfica de Usuario*

4.3.1. Funcionalidad de la GUI

En la siguiente sección se describen las distintas funcionalidades que ofrece la interfaz gráfica de usuario o GUI (*Graphical User Interface*) proporcionada por el Módulo Gráfico.

- Búsqueda rápida

Antes de comenzar una sesión de envío de ficheros, la aplicación permite realizar una búsqueda rápida de dispositivos Bluetooth próximos al equipo Bluetooth. Esto permite que el usuario de la aplicación pueda conocer a priori qué dispositivos participarán posteriormente en la sesión. Tras finalizar la búsqueda, muestra una tabla con información relativa a cada dispositivo encontrado: nombre del dispositivo (Friendly Name), dirección Bluetooth y nombre del usuario (si éste se encuentra registrado).

- Sesiones

Como se ha comentado anteriormente, las sesiones Bluepills permiten realizar el envío de varios ficheros a los dispositivos Bluetooth próximos, y que se encuentren registrados previamente en el fichero de usuarios (ver sección 3.4.1). El diagrama de flujo de la figura 4.8 muestra la secuencia de pasos que debe realizar el usuario al iniciar una sesión Bluepills, a través de la interfaz gráfica. En dicha secuencia, el usuario:

1. Crea una nueva sesión.
2. Selecciona el directorio donde se encuentran los ficheros a enviar.
3. Selecciona el tipo de dispositivos sobre los que realizar el envío (ver siguiente punto “Tipos de sesiones”).
4. Arranca la sesión. Al hacerlo, el Módulo Gráfico invoca al Módulo Bluetooth e inicia una sesión de envío de ficheros (ver sección 4.2.3). Mientras el usuario no detenga la sesión, la aplicación continúa realizando, cíclicamente, nuevas búsquedas de dispositivos no detectados anteriormente. Cada vez que se encuentran nuevos dispositivos Bluetooth, se realiza un nuevo envío de ficheros, si procede (ver siguiente punto “Tipos de sesiones”).
5. Una vez detenida la sesión, el usuario puede opcionalmente guardar los datos de la sesión en un fichero de sesiones (ver sección 3.4.3). En ese caso, el

usuario puede seleccionar actualizar un fichero de sesiones creado anteriormente o solicitar la creación de uno nuevo.

6. A continuación, el usuario puede optar por una de estas dos opciones:
   a) Volver a seleccionar el tipo de dispositivos y arrancar de nuevo la sesión.
   b) Finalizar la sesión.

- Tipos de sesiones

Antes de arrancar una sesión de envío de ficheros, el usuario de la aplicación puede seleccionar el tipo de dispositivos que se incluirán en la sesión:

1. **Usuarios registrados**: usuarios cuyos terminales se encuentren registrados en el fichero de usuarios.
2. **Todos los dispositivos**: todos los dispositivos Bluetooth encontrados que soporten el servicio OBEX Object Push.
3. **Grupo de usuarios**: usuarios registrados que pertenezcan al grupo de usuarios seleccionado (ver sección 3.4.2).
4. **Dispositivos seleccionados**: dispositivos Bluetooth seleccionados por el usuario. Para ello, es necesario que la aplicación haya realizado al menos una primera búsqueda de dispositivos, por lo que esta opción se habilita únicamente tras parar la sesión. Esta opción permite, además, el reenvío de ficheros a cualquier dispositivo.

- Opciones de configuración

La interfaz gráfica de la aplicación Bluepills permite ajustar ciertas opciones sobre la configuración de las sesiones. Estas opciones son:

- Cargar los datos del fichero de usuarios.
- Cargar los datos del fichero de grupos de usuarios.
- Ajustar el tamaño máximo total (en Megabytes) de los ficheros a enviar, contenidos en el directorio seleccionado.
- Ajustar el tiempo de espera (en segundos), durante una sesión, entre nuevas búsquedas de dispositivos.

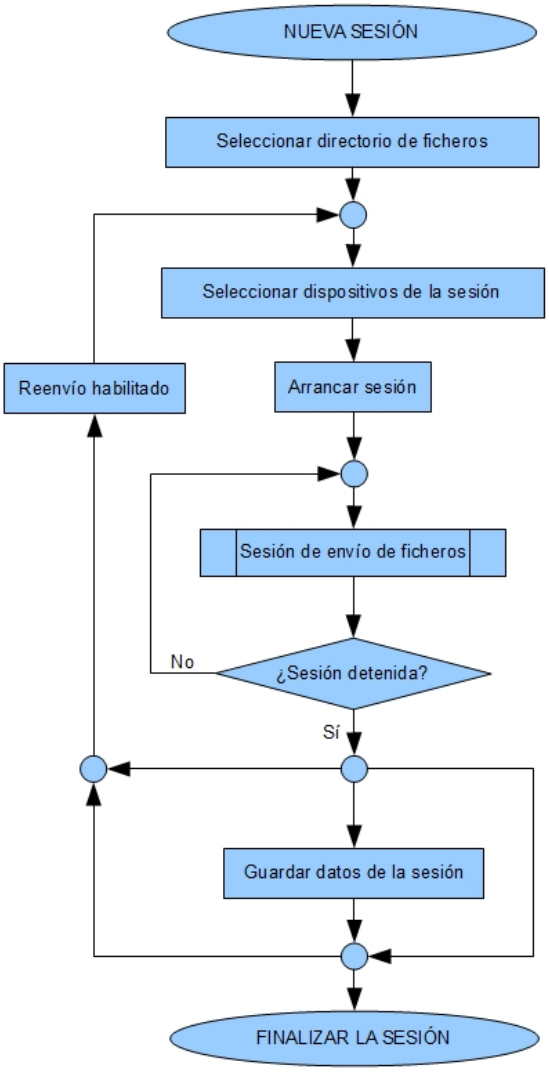


Figura 4.8: *Diagrama de flujo: Sesión a través de la interfaz de usuario*

4.3.2. Estructura de clases del Módulo Gráfico

El Módulo Gráfico está compuesto principalmente por cuatro clases principales y un conjunto de clases auxiliares, contenidas todas ellas en el paquete Java `GUI` (*Graphical User Interface*). La funcionalidad de las clases principales se describe a continuación:

`JFrame_Main`

Constituye la clase principal que contiene el método `main()`, de donde parte el hilo principal de ejecución de toda la aplicación. Construye e inicializa todos los elementos gráficos de la aplicación (ventanas, botones, menús...). Posee múltiples métodos que reaccionan ante eventos de usuario (pulsación de botones, selección de menús...), actuando en consecuencia. Se relaciona con el Módulo Bluetooth invocando sus métodos a través de una instancia de la clase `BTManager`.

`SearchWorker`

Hereda de la clase `SwingWorker` (ver sección 2.4.3), por lo crea un hilo de ejecución independiente que evita el bloqueo de la aplicación. A través de la instancia de la clase `BTManager`, invoca una búsqueda rápida de dispositivos remotos (ver sección 4.3.1). Una vez terminado el proceso de búsqueda, añade los datos de los dispositivos encontrados en una tabla mostrada en la interfaz gráfica.

`SessionWorker`

Al igual que `SearchWorker`, hereda de la clase `SwingWorker`. Utilizando la instancia de la clase `BTManager`, implementa el desarrollo de una sesión Bluepills, a través de sus métodos correspondientes. En función de la interacción del usuario con los botones correspondientes, arranca o detiene la sesión con los parámetros seleccionados (tipo de sesión y opciones de configuración).

`LogWriter`

Imprime mensajes de seguimiento en un cuadro de texto de la interfaz gráfica. Estos mensajes muestran la hora y la acción que va realizando la aplicación en cada momento (búsqueda de dispositivos, envío de ficheros a un

determinado dispositivo Bluetooth, errores...). La instancia de esta clase, por tanto, es invocada oportunamente por el resto de clases de la aplicación.

4.4. Módulo de Datos

4.4.1. Estructura de clases del Módulo de Datos

El Módulo de Datos está constituido por un conjunto de clases Java, contenidas en el paquete `users`, que gestionan las estructuras de datos internas asociadas a los ficheros de datos y que son compartidas por los tres módulos. Para evitar el acceso concurrente o simultáneo a dichos datos (lectura o escritura), lo que provocaría inconsistencia en los mismos, se utilizan métodos sincronizados [15].

La funcionalidad de cada una de las clases de describe a continuación:

`UsersManager`

Gestiona la lectura y escritura de datos relacionados con los usuarios registrados, contenidos inicialmente en el fichero de usuarios. La estructura de dichos datos consiste en un vector de clases de tipo `RegisteredUser`, con una instancia por cada usuario registrado.

`GroupsManager`

Gestiona la lectura y escritura de datos relacionados con los grupos de usuarios, contenidos inicialmente en el fichero de grupos de usuarios. La estructura de dichos datos consiste en un vector de clases de tipo `UsersGroup`, con una instancia por cada grupo de usuarios registrados.

`SessionsFilesManager`

Gestiona la lectura y escritura de datos relacionados con las sesiones Bluepills, que posteriormente pueden almacenarse en un fichero de sesiones. La estructura de dichos datos consiste en un vector de clases de tipo `SessionsFileUser`, con una instancia por cada usuario asociado a las sesiones.

`CsvReader` y `CsvWriter`

Implementan la lectura y la escritura, respectivamente, de cualquier fichero en formato CSV, facilitando el proceso al resto de clases. Ambas clases están contenidas en el subpaquete `users.javacsv`. Pertenecen a la librería *Java CSV* [14], diseñada por Bruce Dunwiddie.

Capítulo 5

Desarrollo de la aplicación Web Bluepills

Como se comentó en el capítulo 3, antes de que el usuario del equipo Bluetooth inicie una sesión de envío de ficheros, es necesario que los usuarios finales se registren en el sistema a través de la aplicación Web de registro. En las siguientes secciones se describen las características y la funcionalidad de dicha aplicación Web, así como su estructura.

5.1. Modelo de la aplicación Web

La aplicación Web Bluepills ha sido desarrollada con la tecnología J2EE (*Java 2 Enterprise Edition*) de Java, utilizando para ello tanto servlets como JSPs (ver sección 2.5). Como servidor de la aplicación Web se ha utilizado *Apache Tomcat 6.0* (ver anexo C).

La aplicación Web Bluepills, como cualquier aplicación J2EE, está compuesta por una serie de componentes Web (servlets, JSPs...) residentes en el contenedor Web del servidor de aplicaciones. El contenedor es un entorno de ejecución que gestiona los componentes, los cuales deben cumplir el contrato que establece el contenedor. Ese contrato no es más que un conjunto de métodos que debe implementar el componente y que permite al contenedor interactuar con él. Además, cada contenedor proporciona una serie de servicios que el componente puede utilizar. El contenedor es además el encargado de gestionar el ciclo de vida de los componentes y realizar la reserva de recursos.

Cada módulo Web dispone de un *descriptor de despliegue* (ver sección C.2.2) que describe cómo se deben desplegar los componentes de dicho módulo dentro del servidor de aplicaciones. Existen además varias formas de empaquetar un módulo Web; en este caso se ha utilizado un archivo WAR (*Web Application Archive*), el cual permite empaquetar en una sola unidad una o varias aplicaciones Web completas, constituidas por servlets, páginas JSP, páginas HTML estáticas, imágenes y otros recursos Web.

En la figura 5.1 se muestra un diagrama de los elementos implicados en el acceso a la aplicación Web. El usuario final, utilizando un navegador Web, accede a la aplicación Web Bluepills a través de la URL correspondiente. El servidor Web, por su parte, responde a través del contenedor de aplicaciones a cada petición del recurso Web solicitado, actuando en consecuencia. En caso de que el recurso sea un documento HTML estático, como ocurre en el acceso a la página principal de la aplicación Web, el servidor devuelve dicho documento. En caso de que el recurso sea un servlet o una página JSP, el contenedor crea un hilo asociado a la instancia del servlet y, a través del método `service()`, determina lo que ha llegado en la petición y devuelve una respuesta dinámicamente, en forma de documento HTML.

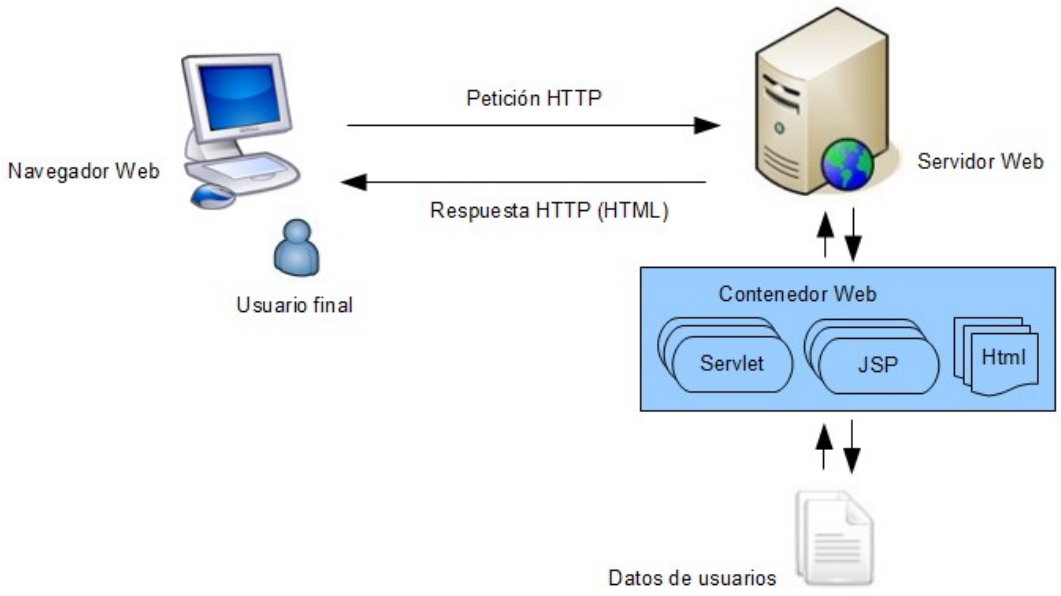


Figura 5.1: *Modelo de la aplicación Web Bluepills*

5.2. Funcionalidad de la aplicación Web

La funcionalidad principal de la aplicación Web Bluepills es la de registrar los terminales móviles de los usuarios finales que, posteriormente, utilizarán el servicio ofrecido por la aplicación Bluepills.

En la imagen de la figura 5.2 se muestra el aspecto que presenta la página principal de la aplicación Web. Cuando un usuario accede a la misma, ésta le permite seleccionar dos perfiles distintos de acceso: de usuario (User Profile) y de administrador (Administrator Profile). Para acceder a la aplicación con cualquiera de ambos perfiles, es necesario introducir una contraseña o clave de acceso, distinta para cada perfil. Dichas contraseñas son configuradas por el gestor de la aplicación Web, a través del descriptor de despliegue asociado (ver sección C.2.2).

En las siguientes subsecciones se analiza la funcionalidad de la aplicación Web desde el punto de vista de ambos perfiles.

Figura 5.2: *Interfaz gráfica de la aplicación Web Bluepills*

5.2.1. Perfil de Usuario

El perfil de usuario permite que los usuarios finales del sistema registren los datos de sus terminales móviles antes de utilizar el servicio Bluepills. Una vez que el usuario ha introducido correctamente su clave de acceso (la cual es común para todos los usuarios del sistema), se solicita al usuario los siguientes datos que serán almacenados en el fichero de usuarios (ver sección 3.4.1): Nombre de Usuario (User Name) y Nombre del Dispositivo (Friendly Name). Aunque el fichero de usuarios también almacena la Dirección Física (Bluetooth Address) del dispositivo móvil, habitualmente el terminal no facilita ese dato al usuario. Es por ello que, posteriormente, cuando se importe el fichero de usuarios al utilizar la aplicación Bluepills, ésta actualizará la Dirección Física del terminal de usuario, una vez encontrado el dispositivo (ver sección 4.2.3). Además, si pasado un tiempo el usuario cambia de terminal pero continúa usando el mismo Nombre de Dispositivo, en la siguiente sesión la aplicación Bluepills actualizará el dato de la Dirección Física del dispositivo. Por otro lado, si el usuario decide cambiar el Nombre del Dispositivo en su terminal, la aplicación Bluepills actualiza ese dato en el fichero. En resumen, puede decirse que siempre y cuando el usuario final conserve alguno de los dos identificadores de su terminal (Dirección Física o Nombre del Dispositivo) la aplicación Bluepills mantiene la coherencia con el fichero de usuarios, actualizándolo convenientemente.

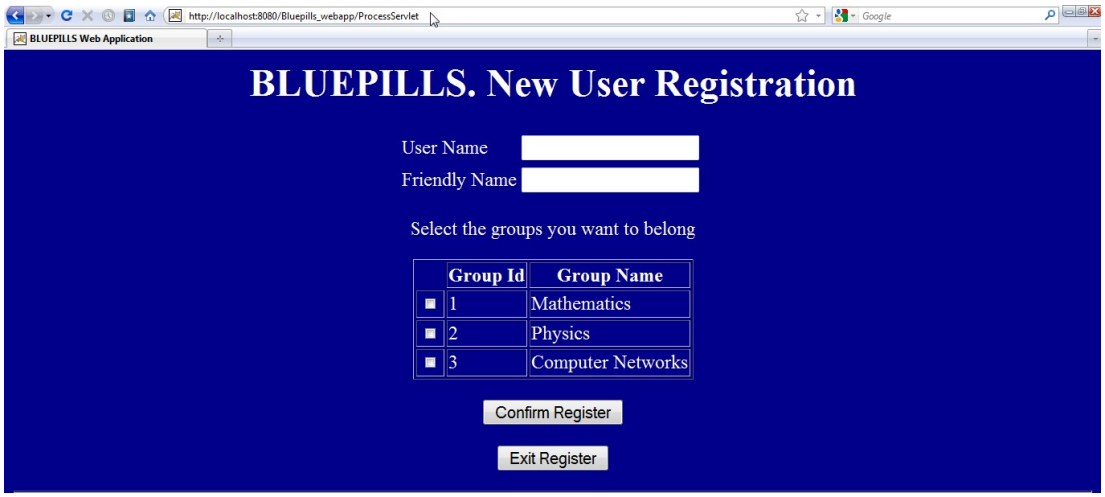


Figura 5.3: *Aplicación Web Bluepills: Perfil de Usuario*

Además de registrar los datos de su terminal, la aplicación permite al usuario inscribirse a uno o más grupos de usuarios. Tras cargar los datos del fichero de grupos, la aplicación Web muestra al usuario una tabla con los nombres de los grupos existentes y sus identificadores (ver sección 3.4.2). El usuario puede marcar los grupos a los que desea pertenecer, marcando uno, varios o ninguno de los grupos.

Tras cumplimentar todos los datos, una vez pulsado el botón "*Confirm Register*" la aplicación Web añadirá los datos del usuario al fichero de usuarios. Si por el contrario el usuario decide cancelar la inscripción pulsando "*Exit Register*", ningún dato quedará registrado. En caso de que alguno de los datos introducidos sea erróneo o esté en blanco, la aplicación muestra un mensaje de error. Si al registrarse un usuario no existe previamente el fichero de usuarios, la aplicación Web lo crea.

5.2.2. Perfil de Administrador

Un usuario con perfil de administrador puede acceder a la información contenida tanto en el fichero de usuarios como en el fichero de grupos. Tras introducir la clave de acceso correspondiente, la aplicación Web muestra un menú con las siguientes opciones al usuario:

- **Registered Users**: Muestra una tabla con la información de los usuarios registrados en el fichero de usuarios. Permite modificar cualquiera de los datos de un usuario o eliminar un usuario.
- **Groups of Users**: Muestra una tabla con la información de los grupos del fichero de grupos. Permite añadir un nuevo grupo, modificar cualquiera de los datos de un grupo o eliminar un grupo.

Si alguno de los datos es introducido incorrectamente, la aplicación Web muestra un mensaje de error. Si al añadir un grupo no existe previamente el fichero de grupos, la aplicación Web lo crea.

5.3. Estructura de la aplicación Web

Como se comentó al principio del capítulo, la aplicación Web Bluepills está compuesta por un conjunto de componentes Web: servlets, JSPs y documentos HTML. Los servlets y JSPs se han utilizado para generar dinámicamente el contenido de las páginas Web de la aplicación, mientras que para páginas Web estáticas (como la página de inicio de la aplicación) se ha utilizado código HTML directamente.

Aunque cualquier aplicación Web puede desarrollarse utilizando exclusivamente servlets o JSPs, suele ser habitual utilizar servlets para procesar los datos de entrada y salida, y JSPs para construir el contenido HTML que se devuelve en cada respuesta al cliente Web. En nuestro caso, ésa ha sido la solución adoptada, y su descripción y funcionamiento se describe en las siguientes subsecciones.

5.3.1. Diseño de servlets

La funciones principales de los servlets dentro de la aplicación Web Bluepills son las siguientes:

- Redirigir las peticiones HTTP de los clientes Web al JSP correspondiente, en función de los parámetros recibidos (pulsación de botones, contenido de los campos en los formularios...).
- Controlar el acceso a los recursos Web de la aplicación, limitándolo a usuarios y administradores autorizados.
- Gestionar el acceso a los ficheros de usuarios y de grupos, tanto de lectura como de escritura.

Cada servlet en Java contiene una única instancia de clase, pero por cada petición HTTP de un cliente distinto se crea un hilo de ejecución independiente. A diferencia de las variables locales de método, que son únicas para cada hilo, los atributos de clase son compartidos por todos los hilos de la instancia. El servidor Web permite el acceso simultáneo de varios clientes (ya sea con perfil de usuario o administrador), por lo que a fin evitar inconsistencia en los datos de los ficheros se ha utilizado la sincronización de hilos proporcionada por la tecnología Java.

Los servlets desarrollados son clases que heredan de la clase `HttpServlet` del paquete `javax.servlet` de Java. Implementan dos métodos principales:

- `init()`: Es invocado al instanciar el servlet cuando el primer usuario accede a la URL del mismo, una vez arrancado el servidor Web. Suele emplearse para inicializar variables o configuraciones del servlet.
- `doPost()`: El método heredado `service()` invoca al método correspondiente en función del tipo de petición HTTP recibida. En el caso de peticiones POST, el método `service()` invoca a este método. En el caso de peticiones GET (petición HTTP por defecto), el método `service()` invocaría al método `doGet()`.

La aplicación Web Bluepills consta de dos servlets, cuya funcionalidad se describe a continuación:

`AccessServlet`

A través de su método `doPost()`, gestiona el acceso a la aplicación Web mediante una clave de acceso (de usuario o administrador) almacenada en el descriptor de despliegue, y que se obtiene previamente en su método `init()`. Una vez verificada dicha clave, el estado de acceso del usuario (permitido o denegado) queda almacenado en una variable de sesión compartida por todos los servlets de la aplicación Web, pero asociada únicamente a ese hilo de usuario. La creación y modificación de variables de sesión se implementa gracias a una instancia de la clase `HttpSession`. Una vez que el usuario sale de la aplicación Web a través de la opción correspondiente, el estado de la variable de sesión cambia a estado denegado.

`ProcessServlet`

Su método `init()` crea las estructuras de datos de usuarios y de grupos, a partir de la lectura de los ficheros correspondientes. La lectura (y escritura) de estos ficheros se realiza a través del paquete `users`, utilizado también en la aplicación Bluepills (ver sección 4.4). El método `doPost()`, por su parte, se encarga de redirigir las peticiones HTTP del cliente Web al JSP correspondiente, pasándole por parámetro los argumentos pertinentes, en función de las acciones realizadas por el usuario. Antes de redirigir la petición, no obstante, se comprueba la autorización de acceso del usuario, a través de la variable de sesión compartida con la clase `AccessServlet`.

5.3.2. Diseño de JSPs

La función principal de los JSPs es la de generar la salida HTML que se devuelve en la respuesta HTTP, de manera dinámica. La aplicación Web Bluepills está constituida por varios ficheros JSP, cada uno de los cuales muestra un contenido distinto en función de la navegación del usuario a lo largo de la misma. La figura 5.4 muestra la relación entre los servlets y los JSPs durante una secuencia de peticiones HTTP del cliente, y que se resume en los siguientes puntos:

1. El usuario, al acceder a la URL de la aplicación a través del cliente Web, recibe como respuesta del servidor la página HTML de inicio: `index.html` (ver figura 5.2). En esta página se solicita al usuario el tipo de perfil y una clave de acceso.

2. Tras seleccionar un perfil, introducir la clave de acceso y seleccionar el botón HTML correspondiente, el cliente Web solicita el recurso servlet `AccessServlet` mediante una petición HTTP POST con los parámetros introducidos.

3. La clase `AccessServlet` verifica el perfil seleccionado y la clave de acceso, y redirige la petición y la respuesta HTTP a la página JSP correspondiente.

4. La página JSP genera dinámicamente el código HTML de respuesta, que devuelve en la respuesta HTTP.

5. Cualquier nueva petición HTTP solicitada a continuación es referenciada al recurso `ProcessServlet`, que se encarga de procesarla.

6. El servlet `ProcessServlet` procesa la petición, realiza las operaciones de entrada y salida pertinentes, y redirige la petición y la respuesta HTTP a la página JSP correspondiente.

7. Si en cualquier momento el usuario selecciona la opción de salir de la aplicación, el servlet `ProcessServlet` redirige la petición y la respuesta HTTP a la página HTML de inicio.

Los ficheros JSP, a diferencia de los servlets, que son clases Java, son documentos planos de texto. Su contenido es similar al de un documento HTML, con la salvedad de que en determinadas zonas se incluyen etiquetas XML especiales encargadas de generar el contenido dinámico de la página. Este formato facilita el diseño gráfico de la aplicación Web pero, a nivel interno, el JSP es traducido por el servidor Web a un servlet. Para más información sobre servlets y JSPs, véase [10].

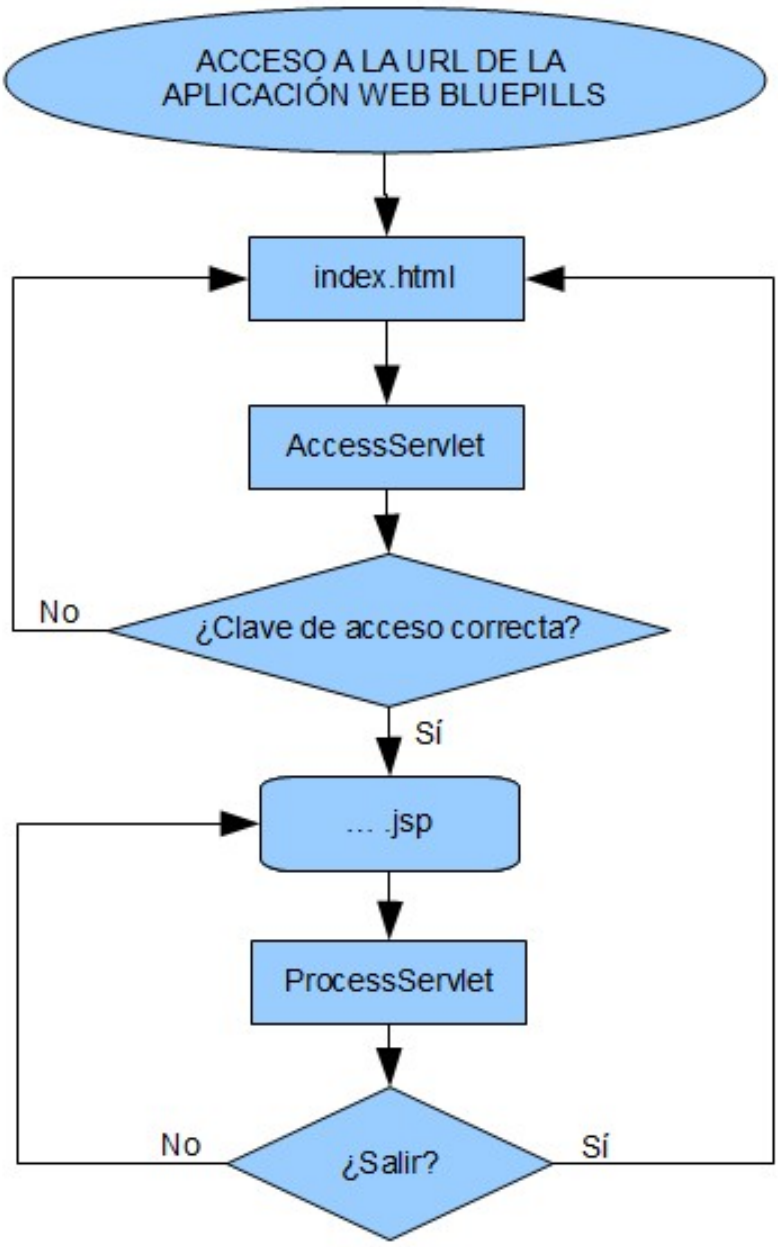


Figura 5.4: *Diagrama de flujo de la aplicación Web Bluepills*

Capítulo 6

Pruebas

En este capítulo se muestra el resultado de las distintas pruebas realizadas al sistema Bluepills. Primeramente, se han realizado pruebas sobre la aplicación Web de registro, generando y modificando datos de los distintos ficheros de usuarios y grupos. A continuación, se han realizado distintas pruebas sobre la aplicación Bluepills, importando los ficheros creados en la aplicación Web, y utilizando distintos terminales móviles.

6.1. Batería de pruebas de la Aplicación Web Bluepills

En el caso de la Aplicación Web Bluepills, se han realizado cuatro tipos de pruebas:

1. Pruebas de creación, modificación y eliminación de grupos de usuarios, utilizando el perfil de Administrador.
2. Pruebas de creación de nuevos usuarios, utilizando el perfil de Usuario.
3. Pruebas de modificación y eliminación de los datos de usuarios existentes, utilizando el perfil de Administrador.
4. Pruebas de acceso simultáneo a la aplicación Web de varios usuarios con distintos perfiles.

En las siguientes tablas se muestran los resultados obtenidos en las distintas pruebas realizadas:

	PRUEBA I: GESTIÓN DE GRUPOS DE USUARIOS
DESCRIPCIÓN	Utilizando el perfil de Administrador, se realizan pruebas de creación, modificación y eliminación de grupos de usuarios.
RESULTADO	Se obtienen resultados satisfactorios en las siguientes pruebas: - Creación de un grupo de usuarios. - Visualización de los datos de los grupos existentes. - Modificación de los datos de un grupo de usuarios existente. - Eliminación de un grupo de usuarios existente.
OBSERVACIONES	- Si el fichero de grupos no existe al añadir un grupo de usuarios, la aplicación lo crea. - Al añadir o modificar un fichero de grupos, la aplicación comprueba previamente que ninguno de los datos coincide con alguno de los grupos ya existentes.

	PRUEBA II: CREACIÓN DE NUEVOS USUARIOS
DESCRIPCIÓN	Utilizando el perfil de Usuario, se realizan pruebas de creación de nuevos usuarios.
RESULTADO	Se obtienen resultados satisfactorios en las siguientes pruebas: - Creación de un nuevo usuario.
OBSERVACIONES	- Si el fichero de usuarios no existe al añadir el nuevo usuario, la aplicación lo crea. - Al crear el nuevo usuario, la aplicación comprueba previamente que ninguno de los datos coincide con alguno de los usuarios ya existentes. - La aplicación carga correctamente los datos de los grupos existentes a los que puede suscribirse el usuario.

	PRUEBA III: GESTIÓN DE USUARIOS
DESCRIPCIÓN	Utilizando el perfil de Administrador, se realizan pruebas de modificación y eliminación de usuarios.
RESULTADO	Se obtienen resultados satisfactorios en las siguientes pruebas: - Visualización de los datos de los usuarios existentes. - Modificación de los datos de un usuario existente. - Eliminación de un usuario existente.
OBSERVACIONES	- Al modificar los datos de un usuario, la aplicación comprueba previamente que ninguno de los datos coincide con alguno de los usuarios ya existentes.

	PRUEBA IV: ACCESO SIMULTÁNEO A LA APLICACIÓN
DESCRIPCIÓN	Utilizando varios equipos conectados a la aplicación Web, se realizan pruebas de creación y modificación simultánea de datos.
RESULTADO	Se obtienen resultados satisfactorios en las siguientes pruebas: - Dos usuarios crean un usuario simultáneamente. - Un usuario crea un usuario mientras que el administrador modifica los datos de otro usuario.
OBSERVACIONES	

6.2. Batería de pruebas de la Aplicación Bluepills

A continuación se muestran los resultados de las distintas pruebas realizadas a la Aplicación Bluepills, cuyos objetivos son:

- Verificar el correcto funcionamiento de la aplicación, en distintos escenarios.
- Comparar las prestaciones obtenidas en terminales de distintos modelos y fabricantes.
- Realizar medidas aproximadas del tiempo de transmisión de los ficheros a los terminales, utilizando ficheros de distintos tamaños.
- Analizar cómo influye la distancia en la transmisión de los ficheros.

	PRUEBA I: SESIÓN CON UN TERMINAL. DISTINTOS FABRICANTES
DESCRIPCIÓN	Sesión de envío de un fichero con un único terminal, de distintos modelos y fabricantes.
ELEMENTOS	**Terminal móvil**: - Nokia 6555 - Nokia 2730 - Sony Ericsson Z750i - Samsung Omnia II **Ficheros**: - Imagen BMP (6KB)
RESULTADOS	- La transmisión del fichero se realiza con éxito en todos los terminales. - En el caso de que el usuario rechace o interrumpa la transmisión, se muestra el pertinente mensaje en la aplicación.
OBSERVACIONES	- El mensaje de solicitud de conexión varía para cada terminal. No obstante, en todos los casos el terminal insta al usuario a escoger entre dos opciones: aceptar o rechazar la recepción del fichero. - En algunos terminales (Nokia), en el caso de que el usuario no responda a la solicitud de conexión, el terminal rechaza la conexión automáticamente, pasados 45 segundos. En otros terminales (Sony Ericsson, Samsung), el terminal continúa a la espera indefinidamente.

	PRUEBA II: SESIÓN CON TRES TERMINALES. DISTINTOS MODOS DE ENVÍO
DESCRIPCIÓN	Sesión de envío de ficheros con tres terminales, utilizando las distintas modalidades de envío. Se utiliza un fichero de usuarios en el que algunos datos de los terminales (nombre del dispositivo o dirección física) difieren de los actuales. También se utiliza un fichero de grupos.
ELEMENTOS	**Terminales móviles**: - Nokia 6555 - Nokia 2730 - Sony Ericsson Z750i **Ficheros**: - Fichero PDF (229KB) - Imagen BMP (6KB) - Fichero TXT (1KB)
RESULTADOS	- Los ficheros se reciben correctamente en los tres terminales. - El fichero de usuarios se actualiza correctamente con los nuevos datos de los terminales. **Varias sesiones con distintas modalidades de envío**: - *Usuarios registrados*: Los ficheros se envían únicamente a los usuarios registrados en el fichero de usuarios. - *Todos los dispositivos*: Los ficheros se envían a todos los dispositivos. - *Grupos de usuarios*: Los ficheros se envían únicamente a los usuarios pertenecientes a un determinado grupo de usuarios.
OBSERVACIONES	- En algunos terminales (Nokia), la solicitud de conexión concierne a todos los ficheros a recibir. En otros terminales (Sony Ericsson), el terminal solicita confirmación al usuario por cada uno de los ficheros.

	PRUEBA III: SESIÓN CON VARIOS TERMINALES. FICHEROS DE GRAN TAMAÑO
DESCRIPCIÓN	Sesión de envío de un fichero de gran tamaño, a uno, dos y tres terminales. Todos los terminales aceptan la recepción del fichero simultáneamente.
ELEMENTOS	**Terminales móviles**: - Nokia 6555 - Nokia 2730 - Sony Ericsson Z750i **Ficheros**: - Fichero MP3 (5MB)
RESULTADOS	- El fichero se recibe correctamente en los terminales, aunque no de manera simultánea. **Tiempo total de recepción del fichero en los terminales (aproximado)**: - 1 terminal: 2 minutos, 30 segundos - 2 terminales: 5 minutos - 3 terminales: 6 minutos, 30 segundos
OBSERVACIONES	- El tiempo de recepción del fichero en cada terminal es variable, aunque ligeramente menor en los terminales Nokia. - En algunos terminales (Nokia), si no hay memoria suficiente para almacenar el fichero, la transmisión no se realiza correctamente y la conexión debe ser interrumpida por el usuario. En otros terminales (Sony Ericsson), el terminal comprueba previamente si dispone de memoria suficiente, antes de iniciar la conexión.

	PRUEBA IV: SESIÓN CON VARIOS TERMINALES. FICHEROS DE TAMAÑO MEDIO
DESCRIPCIÓN	Sesión de envío de varios ficheros a tres terminales. Todos los terminales aceptan la recepción del fichero simultáneamente.
ELEMENTOS	**Terminales móviles**: - Nokia 6555 - Nokia 2730 - Sony Ericsson Z750i **Ficheros**: - Fichero MP3 (2MB) - Imagen BMP (6KB)
RESULTADOS	- Los ficheros se reciben correctamente en los terminales, aunque no de manera simultánea. **Tiempo total de recepción del fichero en los terminales (aproximado)**: - 3 terminales: 2 minutos
OBSERVACIONES	

	PRUEBA V: SESIÓN CON VARIOS TERMINALES. PRUEBAS DE SESIÓN
DESCRIPCIÓN	Sesión de envío de varios ficheros a dos terminales, ejecutando los siguientes pasos: - Se inicia la sesión con único terminal activo. - Tras detener la sesión, se almacenan los datos de la sesión en un nuevo fichero de sesiones. - Se arranca de nuevo la sesión, y se activa el segundo terminal. - Se detiene la sesión y se selecciona el modo de reenvío de ficheros, seleccionando ambos terminales. - Se desactiva el primer terminal y se arranca de nuevo la sesión. - Tras detener la sesión, se selecciona un fichero de sesiones existente y se almacenan los datos de la sesión.
ELEMENTOS	**Terminales móviles**: - Nokia 6555 - Nokia 2730 **Ficheros**: - Imagen BMP (6KB) - Fichero TXT (1KB)
RESULTADOS	Tras ejecutar la secuencia de pasos descrita, se obtiene el resultado satisfactorio esperado.
OBSERVACIONES	- Tras la nueva búsqueda, la aplicación añade el segundo dispositivo a la tabla de de dispositivos, y le transmite los ficheros correctamente. - El reenvío de ficheros se realiza satisfactoriamente sobre el segundo terminal; en el caso del terminal desactivado, la aplicación muestra el pertinente mensaje, al no poder establecer la conexión. - Al guardar el fichero de sesiones existentes, se añaden los datos del segundo terminal.

	PRUEBA VI: SESIÓN CON UN TERMINAL. PRUEBAS DE DISTANCIA
DESCRIPCIÓN	Varias sesiones de envío de ficheros a un terminal, a distintas distancias.
ELEMENTOS	**Terminal móvil**: - Nokia 6555 - Nokia 2730 **Ficheros**: - Fichero MP3 (5MB)
RESULTADOS	**Tiempo total de recepción del fichero en el terminal(aproximado):** - 1 metro: 2 minutos, 30 segundos - 2 metros: 3 minutos - 5 metros: 3 minutos, 20 segundos - 10 metros: 3 minutos, 40 segundos
OBSERVACIONES	- En algunos terminales (Nokia 6555), el alcance máximo es muy reducido: 3 metros. En otros terminales (Nokia 2730), el alcance es de unos 10 metros. - La transmisión de ficheros no se realiza correctamente a través de paredes o muros: la aplicación detecta al dispositivo, pero la conexión no se establece.

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

El objetivo principal del proyecto ha sido el desarrollo de un sistema, denominado Bluepills, que permita el envío de ficheros o píldoras docentes desde el ordenador que utiliza el profesor o ponente a los terminales móviles de los asistentes. Este sistema, desarrollado íntegramente en el lenguaje de programación Java, se ha compuesto a su vez de dos herramientas software: una aplicación gráfica principal y una aplicación Web de registro de usuarios.

Dentro de los objetivos propuestos a priori (ver sección 1.2), se han obtenido los siguientes resultados:

1. A tenor de las pruebas realizadas, se ha podido comprobar que la aplicación Bluepills es compatible con terminales móviles de distintas marcas y modelos, debido a que casi todos los terminales móviles del mercado implementan el servicio OBEX Object Push. No obstante, al ser un estándar abierto, la implementación del perfil OBEX Object Push difiere ligeramente en cada dispositivo, lo cual no perjudica al buen funcionamiento de la aplicación.

2. El sistema resulta ser sencillo y transparente al usuario final, ya que éste no necesita instalar ningún tipo de aplicación o librería en su terminal.

3. La aplicación Bluepills presenta una interfaz gráfica con ventanas, botones y menús que facilitan su uso. Además, al estar desarrollado en Java, es compatible con distintos sistemas operativos.

4. El sistema es capaz de transmitir consecutivamente varios ficheros, de distinto tipo y tamaño.

5. El sistema es capaz de diferenciar a los dispositivos Bluetooth de los asistentes del resto de dispositivos, atendiendo a un determinado tipo de criterio: usuarios registrados, grupo de usuarios o selección manual de dispositivos.

6. La aplicación proporciona un mecanismo de detección de errores en la transmisión de los ficheros, mostrando la información correspondiente, en tiempo real, en un cuadro de texto.

7. La aplicación realiza un seguimiento de los usuarios de cada sesión, almacenando los datos en un fichero.

El uso del perfil OBEX Object Push requiere autorización en lugar de autenticación (ver sección 2.1.6), lo que evita que los usuarios de ambos extremos (equipo y terminal Bluetooth) tengan que introducir una clave de acceso compartida. La autorización, por su parte, asegura que la transferencia de ficheros sólo se realiza tras recibir el consentimiento por parte del usuario final. Además, como se detalla en el desarrollo de la aplicación Bluepills (ver capítulo 4), el proceso de transferencia de datos entre el equipo y cada uno de los terminales Bluetooth se delega en un hilo de Java independiente, lo que permite la transmisión simultánea y evita posibles bloqueos o errores en la aplicación.

Uno de los inconvenientes de utilizar OBEX Object Push frente a otros servicios Bluetooth es que su uso está dirigido a terminales móviles, excluyendo a otros tipos de dispositivos, como ordenadores personales. Una alternativa a OBEX Object Push es OBEX File Transfer, el cual es compatible con ordenadores personales pero, por otro lado, presenta la desventaja de requerir autenticación en su uso.

En lo que a velocidad de transferencia se refiere, el estándar Bluetooth proporciona unos valores inferiores a otros estándares radio de proximidad, como puede ser Wi-Fi. En la siguiente tabla se muestra un resumen de los resultados obtenidos [1] en las pruebas realizadas (ver capítulo 6), al enviar un fichero a varios terminales Bluetooth, de manera simultánea:

Tamaño de fichero	**Terminales**	**Tiempo**	**Velocidad de transferencia**
5 MB	1	150 seg	273 kb/s
5 MB	2	300 seg	273 kb/s
5 MB	3	390 seg	315 kb/s
2 MB	3	120 seg	410 kb/s

Cuadro 7.1: *Velocidad de transferencia en función del número de terminales*

[1] NOTA: Los resultados de la tabla 7.1 se han obtenidos de manera aproximada, tras la ejecución de varias simulaciones

Como podemos observar en los resultados de la tabla anterior, el número de usuarios influye significativamente en la velocidad total de transferencia. En el caso particular de cada terminal, la velocidad de transferencia depende de factores intrínsecos al medio radioeléctrico (ruido, obstáculos...) y de la versión Bluetooth implementada (ver sección 2.1.3). En el caso del equipo Bluetooth, donde reside la aplicación principal, las pruebas han sido realizadas con un dispositivo hardware Bluetooth 2.1. Por tanto, con dispositivos que implementen las últimas versiones del estándar, podrían obtenerse velocidades superiores. Para ficheros más pequeños, los tiempos de transmisión se reducen.

Por otro lado, la distancia entre los dispositivos también influye en la velocidad de transmisión, como muestran los resultados [2] de la tabla 7.2. El alcance máximo es de unos 10 metros, tal y como determina el estándar.

Tamaño de fichero	**Distancia**	**Tiempo**	**Velocidad de transferencia**
5 MB	1 metro	150 seg	273 kb/s
5 MB	2 metros	180 seg	228 kb/s
5 MB	5 metros	200 seg	205 kb/s
5 MB	10 metros	220 seg	186 kb/s

Cuadro 7.2: *Velocidad de transferencia en función de la distancia*

Con respecto a la aplicación Web Bluepills, ésta permite a los usuarios registrar sus terminales móviles en el sistema, antes de utilizar el servicio de envío de ficheros ofrecido por la aplicación principal. Para ello, los usuarios deben conectarse al servidor Web donde reside la aplicación Web a través de Internet o de una red local. Una vez que los datos de la aplicación Web son importados por la aplicación principal, ésta es capaz de detectar a los terminales de los usuarios tanto por el nombre como por la dirección física Bluetooth. Además, la aplicación actualiza ambos datos en cada nueva búsqueda de dispositivos, lo que permite que los usuarios puedan, en cualquier momento, cambiar de nombre o de terminal sin necesidad de realizar un nuevo registro.

En su uso en el ámbito docente, el envío de píldoras docentes a través de Bluepills estaría optimizado para un grupo reducido de usuarios y para ficheros no demasiado grandes. No obstante, dicho escenario no plantea unos requisitos demasiado exigentes en cuanto a velocidad de transmisión, pues la sesión Bluepills puede establecerse paralelamente a la impartición de la clase.

[2]NOTA: Los resultados de la tabla 7.2 se han obtenidos de manera aproximada, tras la ejecución de varias simulaciones

Adicionalmente, el uso de la aplicación Bluepills podría extenderse a otros ámbitos en los que se requiera el envío de pequeños ficheros a dispositivos móviles, de manera local, dada la versatilidad que ésta ofrece.

7.2. Líneas de trabajo futuro

Uno de los aspectos a analizar, en estudios posteriores, es el impacto en la aplicación Bluepills de un número elevado de usuarios. Como se describe en la sección 2.1.2, una red piconet permite un máximo de siete dispositivos esclavos conectados a un único dispositivo maestro. El equipo Bluetooth, al iniciar las conexiones, ejerce de dispositivo maestro mientras que el resto de dispositivos (terminales Bluetooth) ejercen el papel de esclavos. Sin embargo, como se especifica en los parámetros del servicio (ver sección 4.2.4), estos roles pueden intercambiarse, permitiendo el establecimiento de nuevas redes piconet entre el equipo y los terminales Bluetooth. Esta configuración permitiría, teóricamente, establecer más de siete conexiones simultáneas.

Dentro de este mismo estudio, se podría intentar determinar el número óptimo de dispositivos que, conectados simultáneamente, minimizan el tiempo total de transferencia. Una vez hallado este valor, podría incluirse una modificación en el código que evite la transmisión simultánea por encima de ese umbral, manteniendo al resto de dispositivos a la espera.

Otra de las líneas de trabajo futuro podría consistir en desarrollar un nuevo mecanismo de registro de usuarios, alternativo a la aplicación Web Bluepills. Utilizando una aplicación desarrollada en J2ME, los usuarios podrían registrar sus datos a través del terminal móvil, mediante mensajes OBEX/Bluetooth. No obstante, este planteamiento exigiría la transmisión previa de dicha aplicación de registro. En cualquier caso, el uso de un sistema de registro o de seguridad resulta imprescindible, pues garantiza la confidencialidad de los ficheros utilizados en cada sesión.

Apéndice A

Presupuesto

Al final del presente documento se adjunta el presupuesto estimado para la realización del presente proyecto. El proyecto se ha desarrollado en cinco fases, descritas en la sección 1.3. La duración total aproximada ha sido de siete meses.

El presupuesto total del proyecto incluye el coste de los honorarios del Ingeniero y del material empleado. En relación a este último punto, se ha dispuesto del siguiente equipamiento:

- **Ordenador portátil**: Utilizado para el desarrollo íntegro del proyecto, así como para la realización de las pruebas realizadas sobre el sistema desarrollado (ver capítulo 6).
- **Adaptador Bluetooth**: Dispositivo hardware conectado al ordenador pórtatil, el cual permite la comunicación Bluetooth con otros dispositivos.
- **Terminales móviles**: Utilizados para la realización de las pruebas sobre el sistema desarrollado.

Atribuido a otros costes directos del proyecto, se ha incluido el coste de la conexión a Internet durante los siete meses de duración del proyecto, necesaria para la búsqueda de documentación, así como el consumo eléctrico de los equipos. El resto de gastos, como el IVA, se encuentran incluidos dentro de los costes indirectos del proyecto.

Apéndice B

Aplicación Bluepills: Instalación y Uso

B.1. Requisitos

Bluepills es una aplicación desarrollada íntegramente en Java, que utiliza la librería BlueCove (ver sección 2.3.2) para poder utilizar funciones y servicios Bluetooth. Toda su estructura de clases Java se encuentra compilada en un único fichero comprimido: `Bluepills.jar`.

No obstante, antes de ejecutar la aplicación en el equipo, es necesario satisfacer previamente los siguientes requisitos :

- Instalar el entorno de ejecución de Java o JRE (*Java Runtime Environment*), disponible en `http://www.java.com/es/download/`.
- Descargar la librería BlueCove, disponible en `http://bluecove.org/`.
- Disponer de un dispositivo hardware Bluetooth v1.2 o superior.

B.2. Manual de Instalación

Una vez satisfechos los requisitos previos, la aplicación Bluepills no requiere de instalación alguna salvo disponer, en el directorio raíz en el que se encuentre el fichero `Bluepills.jar`, los siguientes subdirectorios:

- \lib: Directorio de librerías, en el que debe incluirse el fichero `bluecove-2.1.0` [1], correspondiente a la librería BlueCove.
- \data: Directorio de datos de usuarios, en el que deben incluirse los ficheros de usuarios y de grupos de usuarios.

B.3. Manual de Uso

B.3.1. Inicio de la aplicación

Para lanzar la aplicación, basta con ejecutar el fichero `Bluepills.jar`. Al hacerlo, se muestra al usuario del equipo una ventana de programa con el siguiente aspecto:

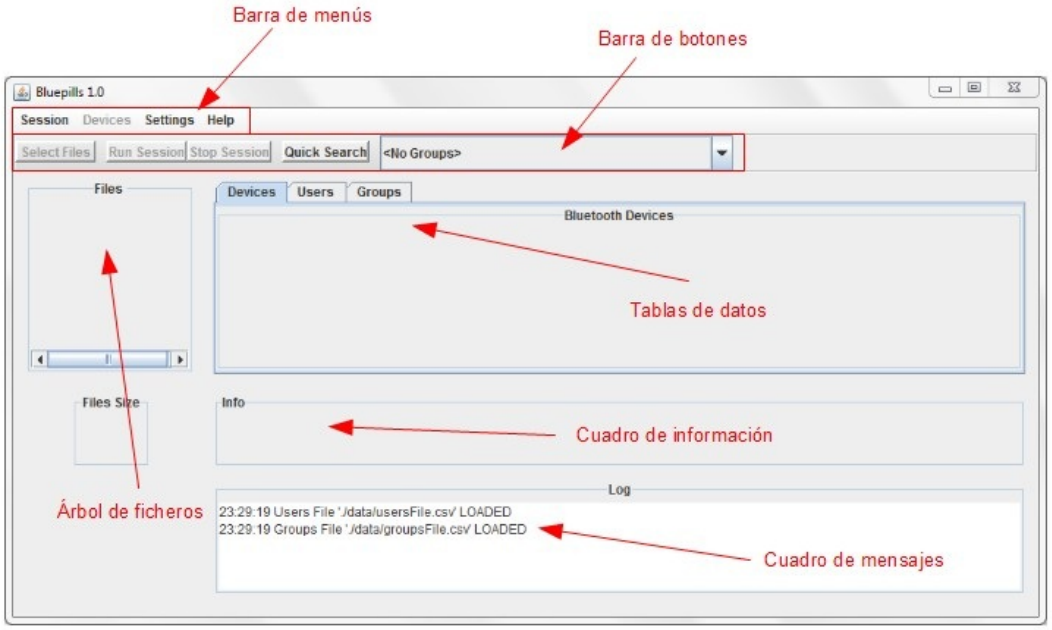


Figura B.1: *Elementos gráficos de la aplicación Bluepills*

[1] NOTA: La librería `bluecove-2.1.0` funciona con cualquiera de las versiones del Sistema Operativo Windows. En el caso de utilizar otros sistemas operativos, consúltese `http://bluecove.org/`

Como se muestra en la figura anterior, la ventana gráfica de la aplicación Bluepills consta de los siguientes elementos:

- **Barra de menús**: Permite gestionar sesiones y modificar los parámetros de las mismas.
- **Barra de botones**: Permite realizar distintas acciones relacionadas con la sesión actual o con la búsqueda rápida de dispositivos.
- **Tablas de datos**: Muestra tres tablas, divididas en pestañas, con información sobre los dispositivos Bluetooth encontrados, los usuarios registrados y los grupos de usuarios.
- **Árbol de ficheros**: Muestra los ficheros a enviar, tras seleccionar un directorio.
- **Cuadro de información**: Muestra un mensaje en tiempo real de la acción que está realizando la aplicación, a nivel de sesión.
- **Cuadro de mensajes**: Lleva un registro de los eventos que van ocurriendo en la aplicación, a lo largo de su ejecución.

Al arrancar la aplicación, ésta trata de cargar los datos del fichero de usuarios y del fichero de grupos, automáticamente. Para ello, estos ficheros deben estar ubicados en el subdirectorio `\data`, y poseer el nombre `usersFile.csv` y `groupsFile.csv`, respectivamente. Si por algún motivo, los ficheros no se cargan correctamente, el usuario puede volver a cargarlos a través del menú "*Settings*". Nótese que para que la aplicación Bluepills funcione, es necesario que ésta disponga de un fichero de usuarios válido. El fichero de grupos, no obstante, es opcional.

Una vez cargados los datos del fichero de usuarios (y de grupos), éstos se muestran en sus correspondientes tablas de datos. A continuación, el usuario del equipo puede realizar una búsqueda rápida de dispositivos Bluetooth, pulsando el botón "*Quick Search*". Una vez terminada la búsqueda, los datos de los dispositivos encontrados se muestran en la correspondiente tabla, lo que permite al usuario conocer a priori, antes de iniciar una sesión (ver sección 4.3.1), qué dispositivos Bluetooth se encuentran a su alcance.

B.3.2. Sesiones en Bluepills

- Configuración de la sesión

A través del menú "*Settings - Options*" pueden configurarse los parámetros de la sesión:

- Tamaño máximo permitido (en megabytes) de todo el conjunto de ficheros a enviar.
- Tiempo de espera (en segundos) entre nuevas búsquedas de dispositivos.

- Inicio de sesión

Para iniciar una sesión, los pasos a seguir son los siguientes:

1. Seleccionar el ítem "*Session - New Session*".
2. Pulsar el botón "*Select Files*" y seleccionar el directorio de ficheros a enviar. Si el tamaño de los ficheros excede el valor de la configuración, muestra un mensaje de advertencia y la sesión no podrá iniciarse.
3. A través del submenú "*Devices - Send Files To*" puede seleccionarse el tipo de dispositivos sobre los que realizar el envío de ficheros:
   - Usuarios registrados (valor por defecto).
   - Todos los dispositivos.
   - Grupo de usuarios. En este caso, el usuario debe seleccionar uno de los grupos de la pestaña de grupos, situada en la barra de botones.
4. Al pulsar el botón "*Run Session*", la sesión de envío de ficheros comienza.

- Gestión de la sesión

Al arrancar la sesión, la aplicación va transmitiendo los ficheros seleccionados a todos los dispositivos Bluetooth circundantes que cumplan el criterio escogido anteriormente. Tras esto, la sesión queda detenida durante el tiempo establecido en la configuración de la sesión. A continuación, la aplicación realiza una búsqueda de nuevos dispositivos no encontrados anteriormente y, en caso de encontrar alguno, lo añade a la sesión y le transmite los ficheros.

En lo que respecta a los usuarios de los terminales móviles, éstos reciben un mensaje de petición de conexión antes de iniciar la transferencia de ficheros vía Bluetooth. Si el terminal no soporta el servicio OBEX Object Push o la solicitud de conexión no logra establecerse, se muestra el siguiente mensaje:

"***ERROR: Connection couldn't be stablished with device***"

Por el contrario, si la solicitud de conexión se recibe correctamente, a continuación pueden aparecer varios tipos de mensajes:

1. "***CONNECTION with device STARTED***": Indica que el usuario final ha aceptado la solicitud de conexión y que la transmisión de ficheros ha comenzado.
2. "***CONNECTION with device completed SUCCESSFULLY***": Indica que la transmisión de ficheros ha concluido satisfactoriamente.
3. "***CONNECTION with device ABORTED***": Indica que el usuario final ha rechazado la solicitud de conexión o que la transmisión de ficheros ha sido interrumpida antes de concluir.

El proceso de búsqueda de nuevos dispositivos se repite cíclicamente hasta que el usuario pulsa el botón "*Stop Session*". Una vez detenida la sesión, el usuario puede observar ciertos parámetros de los dispositivos de la sesión, a través de la correspondiente tabla:

- *Resend*: Indica que la opción de reenvío de ficheros está habilitada.
- *Friendly Name*: Nombre del dispositivo.
- *BT Address*: Dirección Física del dispositivo.
- *User Name*: Nombre del usuario asociado al dispositivo, en caso de estar registrado.
- *Registered*: Indica que el dispositivo se encuentra registrado.
- *Request Sent*: Indica que la solicitud de conexión ha sido enviada al dispositivo.
- *Session Saved*: Indica que la sesión ha sido guardada en el fichero de sesiones.

- *BT Activated*: Indica que el dispositivo continuaba activo en la última búsqueda.

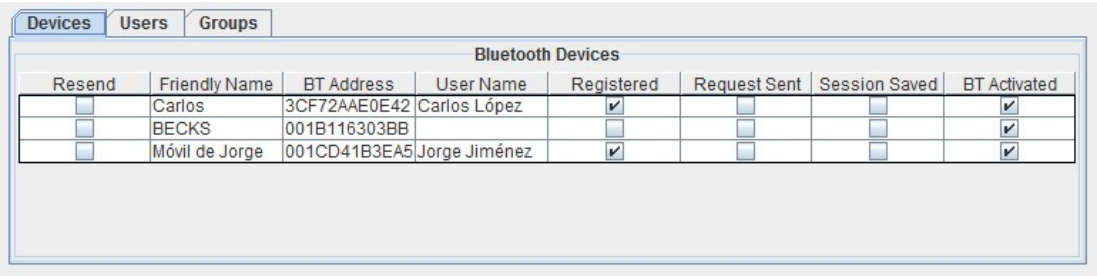


Figura B.2: *Tabla de dispositivos Bluetooth*

A continuación, puede arrancarse de nuevo la sesión pulsando el botón "*Start Session*". Antes de hacerlo, es posible cambiar el tipo de dispositivos de la sesión, seleccionando el submenú "*Devices - Send Files To*". Opcionalmente, puede habilitarse el reenvío de los ficheros a uno o más usuarios de la sesión, seleccionando el submenú "*Devices - Resend Files To - Selected Devices*".

- Guardar datos de la sesión

Estando detenida la sesión, la aplicación permite almacenar los datos de la sesión en un fichero CSV (ver sección 3.4.3). Para ello, el usuario debe:

1. Seleccionar una de las opciones del submenú "*Session - Select Sessions File*":
   a) "*New Sessions File*": Crea un nuevo fichero de sesiones.
   b) "*Existing Sessions File*": Selecciona un fichero de sesiones existentes, al que añadirá la sesión actual.
2. Seleccionar el ítem "*Session - Save Session*": Guarda los datos de la sesión (usuarios, fecha y ficheros enviados) en el fichero de sesiones seleccionado.

- Finalizar la sesión

Para finalizar la sesión, el usuario del equipo debe pulsar el ítem "*Session - End Session*". Al hacerlo, se perderán los datos de la sesión que no hayan sido guardados previamente.

Apéndice C

Aplicación Web: Instalación y Uso

C.1. Manual de Instalación de Apache Tomcat

La aplicación Web Bluepills está diseñada en la tecnología J2EE de Java. Su estructura de ficheros está comprimida en un fichero WAR, que debe ser desplegado en un servidor Web. El servidor Web utilizado en el presente proyecto es el *Apache Tomcat 6.0*, cuyos pasos instalación se describen a continuación:

1. Instalar el entorno de ejecución de Java o JRE (*Java Runtime Environment*), disponible en `http://www.java.com/es/download/`.

2. Descargar la distribución binaria de Apache Tomcat 6.0, disponible en `http://tomcat.apache.org/`, y descomprimirla en el directorio de programas del equipo.

3. Configurar una variable de entorno [1] en el equipo denominada "`JRE_HOME`", y asociarla a la ruta donde se encuentre instalado el JRE.

Una vez instalado el servidor Apache Tomcat, éste se arranca ejecutando el fichero `\bin\startup.bat`. Análogamente, para detener la ejecución del servidor debe ejecutarse el fichero `\bin\shutdown.bat`.

[1] Las variables de entorno en Windows se configuran a través del menú "Configuración del Sistema"

C.2. Manual de Configuración de Apache Tomcat

C.2.1. Estructura de directorios de Tomcat

El directorio raíz donde se encuentra instalado Apache Tomcat consta de los siguientes subdirectorios, los cuales contienen:

- \bin: Ficheros ejecutables de arranque y cierre.
- \conf: Ficheros XML de configuración.
- \logs: Ficheros de registro de eventos y errores.
- \webapps: Aplicaciones Web.
- \work: Ficheros y directorios temporales.

Para que la aplicación Web Bluepills se despliegue correctamente, es necesario ubicar previamente el fichero `Bluepills_webapp.war` en el directorio \webapps de Apache Tomcat, antes de arrancar el servidor.

Una vez arrancado el servidor, es posible acceder al gestor Tomcat a través de la URL `http://localhost:8080/`. Al hacerlo, es necesario introducir un nombre de usuario y contraseña, configurados en el fichero \conf\tomcat-users.xml de la aplicación.

C.2.2. Descriptor de despliegue

El subdirectorio \WEB-INF de la aplicación Web, además de contener los servlets de la aplicación, contiene el fichero `web.xml`, denominado descriptor de despliegue. Este fichero permite, entre otras cosas, declarar servlets y asignarles parámetros. En el caso de la Aplicación Web Bluepills, el descriptor de despliegue contiene los siguientes parámetros configurables:

- *userAccessKey*: Contiene la clave de acceso del perfil de Usuario.
- *adminAccessKey*: Contiene la clave de acceso del perfil de Administrador.
- *usersFilePath*: Contiene la ruta de ubicación del fichero de usuarios.
- *groupsFilePath*: Contiene la ruta de ubicación del fichero de grupos.

C.3. Manual de Uso

C.3.1. Acceso a la aplicación Web

Una vez arrancado el servidor Tomcat, los distintos usuarios de la aplicación Web pueden acceder a la misma a través de la siguiente URL (siendo `<host>` el nombre o dirección IP del servidor):

`http://<host>:8080/Bluepills_webapp`

La página principal de la aplicación Web solicita al usuario que escoja entre perfil de usuario o administrador, y que introduzca su clave de acceso correspondiente.

C.3.2. Perfil de Usuario

El perfil de usuario permite la creación de un nuevo usuario en el sistema Bluepills. Los datos que el nuevo usuario debe introducir son los siguientes:

- **User Name**: Nombre de usuario del sistema Bluepills. Identifica unívocamente al usuario final con un terminal móvil.
- **Friendly Name**: Nombre del dispositivo móvil. Este dato debe ser configurado previamente en las opciones Bluetooth del terminal.
- **Groups**: Grupos a los que el usuario desea pertenecer. Puede seleccionar uno, varios o ninguno.

Una vez introducidos los datos, el usuario debe pulsar el botón "*Confirm Register*" para que los datos queden almacenados en el servidor. Si alguno de los datos introducidos es erróneo o coincide con alguno de los usuarios existentes, la aplicación muestra un mensaje de error y el usuario deberá introducir sus datos de nuevo.

C.3.3. Perfil de Administrador

El perfil de administrador permite visualizar los datos registrados en el sistema Bluepills, así como hacer modificaciones en los mismos. Una vez que el administrador accede al sistema, éste debe seleccionar una de las siguientes opciones de menú:

- **Show Registered Users**: Muestra una tabla con los datos de los usuarios registrados en el sistema. Adicionalmente, el administrador puede modificar o eliminar algunos de los datos de los usuarios.
- **Show Groups of Users**: Muestra una tabla con los datos de los grupos registrados en el sistema. Adicionalmente, el administrador puede modificar o eliminar algunos de los datos de los grupos.

Los datos de usuarios y grupos registrados se encuentran almacenados en dos ficheros CSV, contenidos en la ubicación indicada en los parámetros del descriptor de despliegue (ver C.2.2).

Bibliografía

[1] *Generic Object Exchange Profile*
Bluetooth Specification v1.1 2001.
http://www.bluetooth.com/SiteCollectionDocuments/GOEP_SPEC_V12.pdf

[2] *Object Push Profile*
Bluetooth Specification v1.1 2001.
http://www.bluetooth.com/SiteCollectionDocuments/OPP_SPEC_V11.pdf

[3] *BlueCove*
http://bluecove.org

[4] *Bluetooth Application programming with the Java APIs*
Morgan Kaufmann Publishers
KUMAR C Bala, KLINE P.J., THOMPSON T.J., 2004.

[5] *PFC: Seguridad en Bluetooth*
Universidad Pontificia Comillas
Alberto Moreno Tablado, 2006.

[6] *Programación de dispositivos Bluetooth a través de Java*
JavaHispano. Alberto Gimeno Brieba
http://www.javahispano.org/contenidos/es/jsr82_bluetooth_desde_java/

[7] *JSR 82 Bluetooth API and OBEX API*
http://download.oracle.com/javame/config/cldc/opt-pkgs/api/bluetooth/jsr082/index.html

[8] *Swing y JFC (Java Foundation Classes)*
Programación en castellano
http://www.programacion.com/articulo/swing_y_jfc_java_foundation_classes_94

[9] *The Java tutorials: Graphical User Interfaces*
http://download.oracle.com/javase/tutorial/ui/overview/intro.html

[10] *Beginning J2EE 1.4: From Novice to Professional*
Apress
WEAVER James L., 2004

[11] *Servlets y JSP*
http://distritos.telepolis.com/java/lib/manuales/servletsjsp.pdf

[12] *JavaTM 2 Platform Enterprise Edition 5.0 API Specification*
http://download.oracle.com/javaee/5/api/

[13] *JavaTM 2 Platform Standard Edition 5.0 API Specification*
http://download.oracle.com/javase/1.5.0/docs/api/

[14] *Librería Java CSV 2.1*
Bruce Dunwiddie
http://www.csvreader.com

[15] *Sincronización de hilos en Java*
Programación en castellano
http://www.programacion.com/articulo/threads_de_control_100/14

[16] *Introducción a la tecnología Bluetooth*
Kioskea.net
http://es.kioskea.net/contents/bluetooth/bluetooth-intro.php3

[17] *Tecnología Bluetooth*
Monografías.com
http://www.monografias.com/trabajos43/tecnologia-bluetooth/tecnologia-bluetooth2.shtml

[18] *Diferencia entre las versiones de Bluetooth*
Bucéfalo. Tecnología y actualidad
http://bucefalo.com.mx/diferencia-entre-las-versiones-de-bluetooth

A continuación se anexa el presupuesto desglosado del proyecto, el cual asciende a la cantidad de 23.297 euros.

Leganés, a 6 de julio de 2011

El ingeniero proyectista

Jorge Beca Baulenas

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:

Jorge Beca Baulenas

2.- Departamento:

Ingeniería Telemática

3.- Descripción del Proyecto:

- Titulo: **Bluepills: Envío de píldoras docentes a través de Bluetooth**
- Duración (meses): **7**
- Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros):

23.297 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
		Ingeniero Senior		4.289,54	0,00	
Beca Baulenas, Jorge		Ingeniero	7	2.694,39	18.860,73	
		Hombres mes	**7**	**Total**	18.860,73	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Ordenador portátil - HP Pavilion	800,00	100	7	60	93,33
Adaptador Bluetooth -Trust BT 2400P	12,00	100	7	60	1,40
Teléfono móvil - Nokia 6555	150,00	100	7	60	17,50
Teléfono móvil - Nokia 2730	80,00	100	1	60	1,33
Teléfono móvil - Sony Ericcson Z750i	120,00	100	1	60	2,00
				Total	115,57

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} \text{xCxD}$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Consumo eléctrico	Unión Fenosa	52,5
Conexión a Internet	Movistar	385,00
	Total	437,50

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas,

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	18.861
Amortización	116
Subcontratación de tareas	0
Costes de funcionamiento	438
Costes Indirectos	3.883
Total	23.297